Universidad Carlos III de Madrid
Escuela Politécnica Superior
Departamento de Informática
Avda. de la Universidad, 30
28911, Leganés
Madrid, España

**Referencia:** PFC_Esteban_Cobo_Ceballos
**Versión:** 1.A
**Fecha:** 21/06/2013

# PROYECTO DE FIN DE CARRERA

# Diseño e Integración en Android de un Sistema de Realidad Aumentada y Reconocimiento de Imágenes para un Sistema de Domótica Asistencial

**Titulación: Ingeniería Técnica en Informática de Gestión**

| | Nombre | Firma | Fecha |
|---|---|---|---|
| Autor | Esteban Cobo Ceballos | | |
| Tutor | Javier Fernández Muñoz | | |
| Co-director | Alberto Jardón Huete | | |

Fichero: PFC_Esteban_Cobo_Ceballos.doc

**Título:** Diseño e Integración en Android de un Sistema de Realidad Aumentada y Reconocimiento de Imágenes para un Sistema de Domótica Asistencial.

**Autor:** Esteban Cobo Ceballos.

**Tutor:** Javier Fernández Muñoz.

**Co-director:** Alberto Jardón Huete

**EL TRIBUNAL**

**Presidente:** _____

**Vocal:** _____

**Secretario:** _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

**VOCAL**

**SECRETARIO** **PRESIDENTE**

## Resumen

Este proyecto describe el desarrollo de una aplicación Android para poder manejar los electrodomésticos de una casa domótica a través del reconocimiento de imágenes. Los usuarios podrán con tan solo enfocar la cámara del móvil a la imagen ejecutar las órdenes al electrodoméstico mediante su voz. El presente documento tiene como finalidad la presentación del proceso completo llevado a cabo para la realización del proyecto.

## Abstract

This project describes the development of an Android application that allows manages the devices in a domotic house by image recognition. The users of the system will be able of executing commands using the phone camera and the voice. The current document details the complete process needed to create the project.

## Lista de Distribución

| Interna | Copias | Externa | Copias |
|---|---|---|---|
| Universidad Carlos III de Madrid | 1 | | |

## Registro de Cambios

| Versión | Fecha | Sección / Página | Descripción del Cambio |
|---|---|---|---|
| 1.A | 21/06/2013 | Todo | Primera versión |

# Índice de Contenidos

# Índice de Figuras

## Índice de Tablas

# 1. INTRODUCCIÓN

Esta introducción esta enfocada a acercar al lector a las motivaciones, objetivos y contenidos que conforman las bases de este proyecto. A continuación se detallan cada una de los mencionados temas.

## 1.1 VISIÓN GENERAL

El mundo de la telefonía móvil ha evolucionado enormemente en los últimos tiempos, de tal manera que hemos pasado, en tan solo un par de décadas, de manejar dispositivos móviles con limitadas funcionalidades y poco prácticos en cuanto a usabilidad se refiere, a maquinas con infinidad de recursos y de reducido tamaño, que abren un gran abanico de posibilidades de uso. Actualmente tanto la capacidad y rendimiento, como las diferentes plataformas que existen para dispositivos móviles posibilitan la creación de un ilimitado número de aplicaciones que cubren cualquier tipo de necesidad que pueda tener el usuario de un dispositivo móvil.

Por otro lado con motivo de los avances informáticos sufridos en los últimos años, el sector eléctrico no quedándose atrás y aprovechando el uso de estos avances, con la tecnología ya existente, se plantea la idea de centralizar las órdenes y las funciones del uso doméstico con el fin de evitar tareas repetitivas e incómodas. La tecnología ha marcado un desarrollo creciente en todo el mundo, llegando a un nivel tal que una casa pueda realizar por ella misma sus tareas domésticas, además de contener un sistema integrado de seguridad, ventilación, alumbrado…

De aquí surge el concepto de domótica, que es la integración de la tecnología inteligente, que permite automatizar una vivienda, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. Las posibilidades de la domótica son innumerables y permiten un paquete de servicios absolutamente personalizado y adaptado a los habitantes de la vivienda.

Este proyecto surge de la idea de unir estas dos tecnologías, la de los dispositivos móviles de alta generación y las casas domóticas o inteligentes, para crear un sistema móvil que permita el manejo de electrodomésticos a personas impedidas o con movilidad reducida. El usuario de esta manera podrá, con solo enfocar la cámara del móvil a una etiqueta o imagen, ejecutar las órdenes al electrodoméstico mediante su voz.

## 1.2 MOTIVACIÓN

La principal motivación del proyecto es la de ofrecer acceso a usuarios con movilidad reducida a los diferentes servicios que ofrece la tecnología de la domótica. Actualmente la

forma de interactuar de un electrodoméstico es puramente física, lo que implica en personas impedidas un nivel de dependencia muy grande. A través de este proyecto podrán utilizar electrodomésticos y otros servicios de la casa domótica, de tal manera que pueda aumentar su nivel de independencia en su vida diaria.

## 1.3 OBJETIVOS

Como se ha comentado anteriormente, el objetivo general del proyecto es la creación de una aplicación móvil que permita reconocer una imagen o etiqueta mediante la cámara y manejar un electrodoméstico de la casa domótica mediante voz.

Los objetivos técnicos del proyecto que permiten la consecución de este objetivo general se podrían resumir en los siguientes:

- Desarrollo de una aplicación móvil mediante la plataforma Android.
- Reconocimiento de imágenes o etiquetas mediante la cámara del móvil.
- Utilización de un sistema de realidad aumenta para mostrar información “aumentada” al usuario.
- Interactuar con el sistema de domótica existente para la ejecución de controles sobre los dispositivos.
- Proveer interfaces de voz al usuario, tanto síntesis de voz (*Text To Speech*) como reconocimiento de voz.

## 1.4 ESTRUCTURA DEL DOCUMENTO

La memoria de este proyecto está estructurada en 12 capítulos, que a su vez se subdividirán en otros apartados. A continuación se describe el contenido de cada uno de estos capítulos.

**Capítulo 1 Introducción**

Pretende dar una visión global del proyecto, introduce los objetivos y las motivaciones para ejecutar el proyecto, así como la estructura de este documento.

**Capítulo 2 Estado del Arte**

Se analiza la evolución de la tecnología en diferentes ámbitos y, en base a ello y a las expectativas del proyecto, se realiza la elección de plataforma de desarrollo y de las tecnologías a utilizar en la aplicación.

**Capítulo 3 Análisis**

Mostrará el conjunto de los requisitos y especificaciones concretas que son necesarias para poder llevar a cabo correctamente el proyecto.

**Capítulo 4 Diseño**

Este capítulo describe las decisiones de diseño tomadas para el desarrollo de procesos, interfaces y de implementación.

**Capítulo 5 Implementación**

Se explica de forma resumida la utilización de APIs externas al proyecto y su uso dentro del código fuente de la aplicación.

**Capítulo 6 Pruebas**

Especifica las diferentes situaciones controladas a las que se someterá el sistema para comprobar el correcto funcionamiento, así como los resultados obtenidos.

**Capítulo 7 Gestión del Proyecto**

Detalla la planificación que se quiere seguir a lo largo de las distintas fases del proyecto, la metodología utilizada y el presupuesto del sistema completo de acuerdo con la duración estimada.

**Capítulo 8 Conclusiones y Líneas Futuras**

Contiene una breve conclusión obtenida como consecuencia de la finalización del proyecto, así como los posibles trabajos que se podrían realizar sobre este proyecto fin de carrera como extensión al mismo.

**Capítulo 9 Glosario**

Este capítulo describe la lista de los acrónimos y palabras de difícil comprensión para el lector contenidas en el proyecto.

**Capítulo 10 Bibliografía**

Muestra las referencias documentales que han sido utilizadas durante la documentación del proyecto.

**Capítulo 11 Anexo A: Manual de Instalación**

Contiene la lista de instrucciones que se deberán llevar a cabo para poder instalar la aplicación.

**Capítulo 12 Anexo B: Manual de Usuario**

Destinado al usuario final de la aplicación móvil.

**Capítulo 13 Anexo C: Guía de Despliegue**

Contiene la lista de instrucciones que se deberán llevar a cabo para poder desplegar completamente el sistema desde el código fuente.

# 2. ESTADO DEL ARTE

En este capítulo se dará al lector una visión ampliada del estado actual de las tecnologías utilizadas para la realización de este proyecto. Cinco apartados claramente diferenciados se describen: dispositivos móviles y Android; síntesis y reconocimiento de voz; realidad aumentada; visión artificial; y por último sistemas de domótica.

Al finalizar este capítulo se debe de tener claro las tecnologías o frameworks a utilizar en el desarrollo del proyecto.

## 2.1 DISPOSITIVOS MÓVILES Y ANDROID

### 2.1.1 Dispositivos Móviles

Por dispositivo móvil se entiende cualquier computadora de tamaño reducido con capacidad de procesamiento, conexión a Internet permanente o intermitente y un tamaño de memoria limitado. Algunos de los dispositivos móviles más comunes son:

- Smartphone (teléfonos inteligentes de última generación).
- PDA´s.
- Celulares.
- Videoconsola portátil.
- Reproductor de audio portátil.
- Cámara digital.
- Cámara de video.
- Laptop (portátil).

Los controles de dichos dispositivos pueden ser o bien pequeños botones o como ya sucede en la actualidad, pantallas táctiles o bien híbridos de ambos tipos, táctiles y con botones.

**Figura 2-1: Ejemplos de dispositivos móviles**

De entre todos ellos, los que más relevancia tienen en la actualidad, son los denominados "*smartphones*" o teléfonos inteligentes. Estos dispositivos son teléfonos móviles que incorporan características propias de un ordenador personal. Permiten la instalación de aplicaciones, tienen conexión a Internet, y otras muchas funcionalidades, todo ello gracias a que normalmente incorporan un potente sistema operativo como por ejemplo "*Android", "IOS", "Symbian OS", "BlackBerry OS",* etc…, Algunos de ellos será estudiados más extensamente en apartados sucesivos.

### 2.1.2 Sistemas Operativos para Dispositivos Móviles

Actualmente hay un gran abanico de sistemas operativos diseñados específicamente para funcionar en dispositivos móviles. Estos sistemas operativos han sido concebidos para obtener el mayor rendimiento a partir de una limitación de recursos, como suele ocurrir en la mayoría de dispositivos móviles. Ya que estos suelen contar con funcionalidades propias de ordenadores personales pero no así con la capacidad que ellos tienen, en cuanto a memoria y capacidad de procesamiento se refiere.

Los más conocidos del mercado son los siguientes:

- Android
- Symbian OS
- iOS
- BlackBerry OS
- Windows Phone

- Otros:
  - Bada
  - MeeGo

A continuación se muestra un gráfico con el porcentaje de implantación de los diferentes sistemas operativos durante 2013. Como se puede observar, en la actualidad los sistemas operativos que acaparan el mercado son "*Android*" e "*IOS*", entre ambos aglutinan más del 90 % de la cuota de mercado. El sistema operativo "*Android*" de "*Google inc*" se expondrá más a fondo en siguientes apartados.

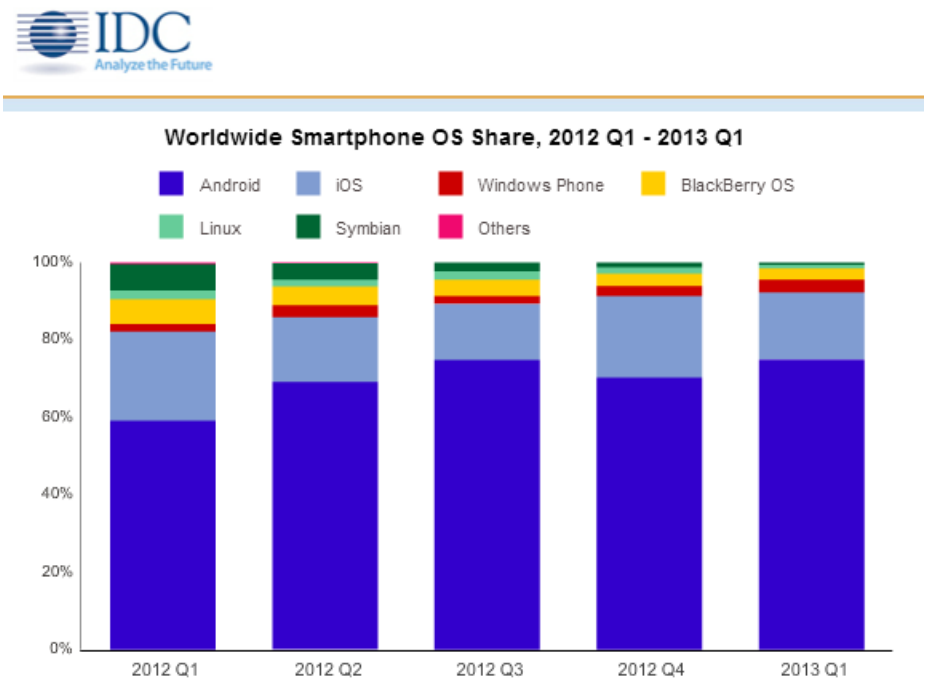


**Figura 2-2: Cuota de mercado de smartphones por Sistema Operativo en 2013**

### 2.1.2.1 Sistema operativo Android

El sistema operativo "*Android*", propiedad de "*Google inc*", es en la actualidad el sistema operativo para smartphones que más cuota de mercado acapara, aunque también existen versiones del mismo para "*tablets*", estamos hablando de las versiones 3.X denominada "*HoneyComb*".

En la actualidad la versión de Android predominante en el mercado de los smartphones, es decir, la más implantada por las distintas compañías de telefonía móvil en sus dispositivos, es la versión 2.3.X ó "*Gingerbread*", según los datos estadísticos basados en los terminales que se conectan a "*Android Market*" (sitio web para descarga de aplicaciones para dispositivos móviles Android) publicados por Google, con un total del 36,4%, aunque ya le sigue muy de cerca la versión 4.1.X ó "*Jelly Bean*" con un 29,0%. La figura mostrada a continuación muestra un gráfico con los datos publicados por Google, con respecto a la

implantación de las distintas versiones de Android en dispositivos móviles (basado en los terminales que se conectan a “Android Market”).

| Version | Codename | API | Distribution |
|---|---|---|---|
| 1.6 | Donut | 4 | 0.1% |
| 2.1 | Eclair | 7 | 1.5% |
| 2.2 | Froyo | 8 | 3.2% |
| 2.3 - 2.3.2 | Gingerbread | 9 | 0.1% |
| 2.3.3 - 2.3.7 | | 10 | 36.4% |
| 3.2 | Honeycomb | 13 | 0.1% |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 25.6% |
| 4.1.x | Jelly Bean | 16 | 29.0% |
| 4.2.x | | 17 | 4.0% |

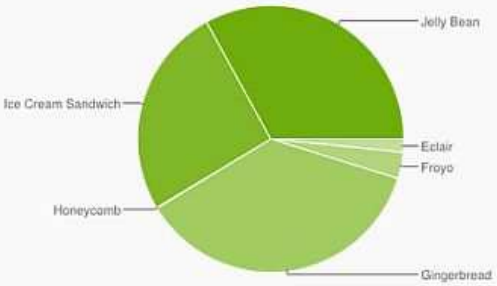


**Figura 2-3: Implantación de las distintas versiones de Android (2013)**

La implantación de este sistema operativo en dispositivos móviles ha sido exponencial desde su irrupción en el mercado de la telefonía móvil por 2008 [1], su popularidad se debe en parte a la pertenencia de Google a la “*OHA*” (“*Open Handset Alliance*”), alianza comercial entre 84 compañías pertenecientes al mundo de las telecomunicaciones, entre las que se encuentran algunos de los mayores fabricantes de dispositivos móviles a nivel mundial, como pueden ser *HTC, Samsung, LG, Motorola, Sony Ericsson, etc…*, que han incluido Android en prácticamente todos los nuevos modelos que han ido sacando al mercado. Esto es debido en parte a una de las principales características de Android, es de código abierto, y cada una de las empresas mencionadas anteriormente, ha podido modelar el sistema operativo conforme a sus necesidades.


**Figura 2-4: Miembros pertenecientes a la “Open HandSet Alliance”.**

## 2.1.3 Plataforma Android

### 2.1.3.1 Arquitectura del sistema Android

La arquitectura "*Android*" esta basada en el kernel de Linux, es por ello, que tiene un núcleo que aporta a este sistema operativo rapidez, fiabilidad y seguridad probadas.

Internamente, Android utiliza Linux para la gestión de su memoria, de los procesos, para las operaciones de red y para otros servicios del sistema operativo. No obstante, el usuario nunca interaccionará directamente contra Linux, tampoco lo harán las aplicaciones instaladas en el dispositivo móvil, será pues el propio "*Android*", el que interactúe con el. A continuación, se muestra en la imagen, la estructura que sigue un sistema "*Android*".


**Figura 2-5: Arquitectura sistema Android**

La arquitectura "*Android*" implementa las capas que se describen a continuación:

- Bibliotecas nativas:

  Es la capa situada por encima del Kernel y contiene las bibliotecas nativas de "Android". Estas bibliotecas compartidas están implementadas en C ó C++, y compiladas para la arquitectura de hardware específica utilizada por el dispositivo móvil y preinstaladas por el fabricante del mismo.

  Algunas de las bibliotecas nativas de Android de mayor trascendencia son las siguientes:

  - Gestor de superficie: es el gestor de ventana de composición utilizado por "Android". Permitirá al sistema crear todo tipo de efectos como ventanas transparentes.
  - Gestor de gráficos 2D y 3D: en "Android", es posible combinar en una sola interfaz de usuario elementos de 2 y 3 dimensiones, y esta biblioteca permitirá al sistema tomar la decisión de utilizar hardware 3D si es que el dispositivo móvil dispone de el, o un renderizador de software en caso contrario.
  - Códecs multimedia: "Android" puede reproducir video y grabar o reproducir audio, esta librería contiene los codecs necesarios para reproducir un determinado elemento multimedia en distintos formatos (AAC, AVC, MP3 y MP4).
  - Base de datos SQL: "Android" incluye un motor de bases de datos "SQLite", la misma base de datos utilizada por el navegador "Firefox" y el "iPhone" de "Apple". La utilizarán las aplicaciones "Android" para almacenamiento persistente.
  - Tecnología de navegador: utilizada por el sistema para la muestra rápida de contenido HTML, Android utiliza la biblioteca "WebKit" [2] para este cometido.

- Tiempo de ejecución:

  También sobre la capa del Kernel se encuentra la capa de tiempo de ejecución, que está provista de la maquina virtual "*Dalvik*".

  Esta maquina virtual tiene como especial peculiaridad, que el código se compila en ella por medio de instrucciones independientes denominadas "*bytecodes*", que son ejecutadas por la propia maquina virtual en el dispositivo móvil.

  Esta maquina virtual desarrollada por Google, realmente es una maquina virtual Java, optimizada para su utilización en dispositivos con restricciones de memoria.

  Todo el código escrito en Java para un sistema "*Android*" se ejecutará en esta maquina virtual.

- Estructura de aplicaciones

  Esta capa esta situada por encima de las capas de "*bibliotecas nativas*" y de la capa de "*tiempo de ejecución*". Esta capa contiene los bloques de construcción de alto nivel en los que nos apoyaremos a la hora de crear aplicaciones para el sistema "*Android*". Los componentes más importantes de la estructura de aplicaciones son los siguientes:

  - Gestor de actividad: este gestor es el encargado de controlar el ciclo de vida de las aplicaciones y mantiene un orden entre las aplicaciones, para que el usuario pueda moverse de unas a otras.
  - Proveedor de contenido: estos objetos encapsulan datos que tienen que ser compartidos entre distintas aplicaciones.
  - Gestor de recursos: encargado de gestionar cualquier tipo de elemento de la aplicación que no sea código.
  - Gestor de ubicación: se encarga de posicionar el dispositivo móvil en todo momento.
  - Gestor de notificaciones: es el encargado de la gestión de cualquier tipo de notificación que tenga que mostrarse al usuario, como por ejemplo, citas, alertas, etc…

- Aplicaciones y widgets

  Esta es la capa superior en el diagrama de arquitectura de un sistema "*Android*", y será la que contenga todas las aplicaciones y widgets (aplicaciones que ocuparan una pequeña parte de la pantalla de inicio y toda su funcionalidad se podrá llevar acabo desde la misma) contenidos en nuestro dispositivo móvil, tanto los preinstalados en el como los instalados posteriormente por el usuario.

### 2.1.3.2 Ciclo de vida de un Actividad

La siguiente imagen muestra el ciclo de vida de una Actividad o *Activity*, que es el componente principal de una aplicación Android.

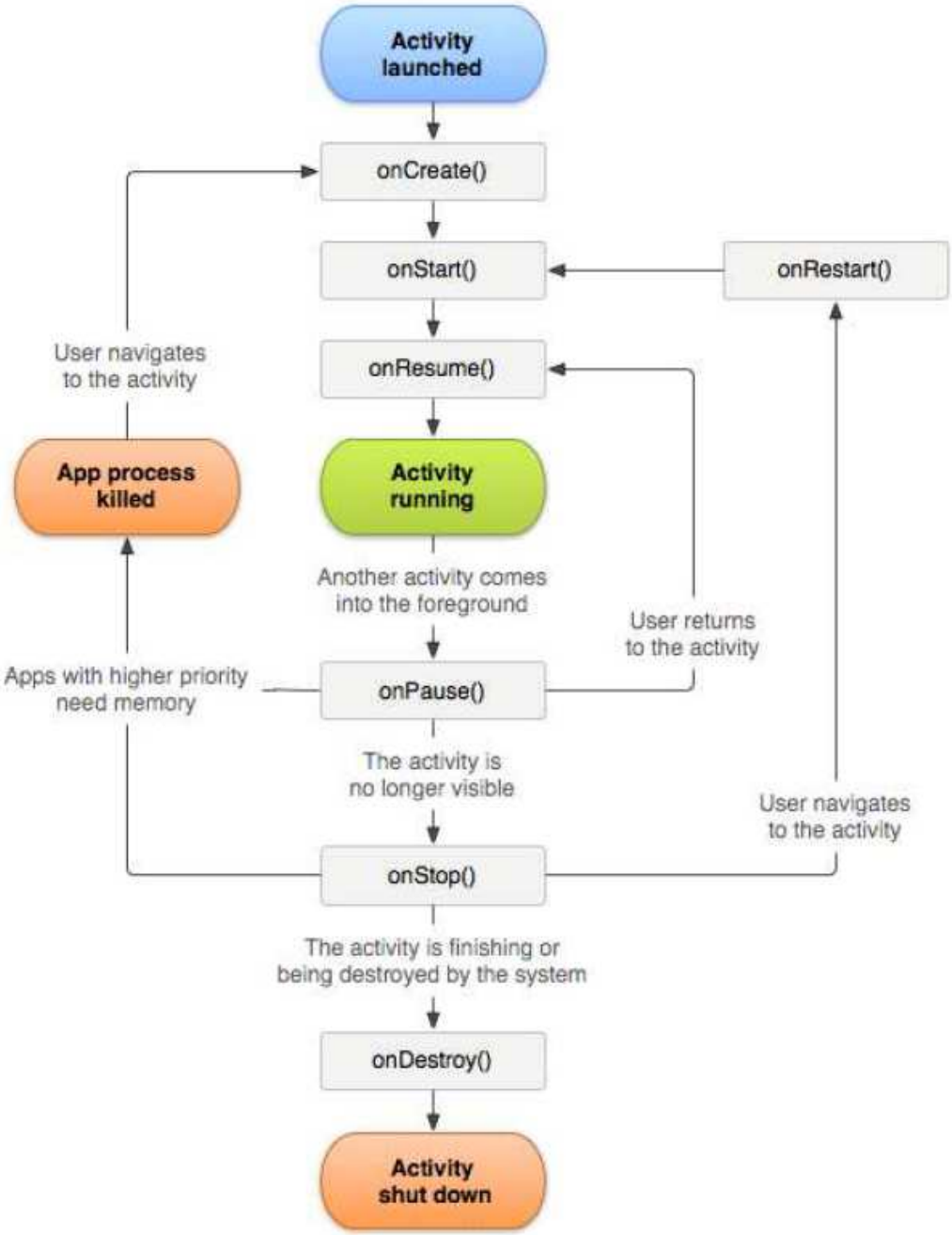


**Figura 2-6: Ciclo de vida de una actividad Android**

El ciclo de vida de una aplicación "*Android*" es algo diferente al de una aplicación convencional. Internamente, cada pantalla de la interfaz de usuario esta representada por una clase "*Activity*", y cada una de ellas tiene su propio ciclo de vida dentro de la plataforma "*Android*". A su vez, una aplicación "*Android*" esta compuesta por una o varias actividades y un proceso *Linux* que las contiene. El ciclo de vida del proceso contenedor no esta directamente relacionado con el ciclo de vida de las actividades de una aplicación, que puede estar "viva" aunque el proceso Linux haya dejado de ejecutarse. Por ejemplo en el caso de que la aplicación se quedase en un segundo plano de ejecución. Este mecanismo se utiliza para liberar recursos del sistema, cuando una determinada aplicación no está en uso.

El ciclo de vida de una actividad "*Android*" esta compuesto por diferentes estados de ejecución de la misma. El cambio entre unos u otros estados no está al alcance de los desarrolladores, será una función que solo podrá llevarse acabo por el sistema operativo. No obstante, este se encarga de notificar el cambio de un estado a otro de una determinada actividad mediante la ejecución de los métodos "*onXX*" que ya si que podrán ser implementados en cada actividad por el desarrollador, para tener un cierto control sobre los cambios de estado durante la ejecución de la actividad. A continuación se especifican los métodos de que dispone el desarrollador para manejar dichos cambios de estado:

- onCreate(Bundle):

  Este método es ejecutado cuando se inicia la actividad por primera vez.
- onStart():

  Este método es ejecutado cuando va a mostrarse la actividad al usuario.
- onResume():

  Es ejecutado cuando la actividad puede empezar a interactuar con el usuario.
- onPause():

  Se ejecuta cuando una actividad va a pasar a ser ejecutada en segundo plano.
- onStop():

  Se ejecuta cuando la actividad ya no esta siendo visualizada por el usuario y no va a ser ejecutada durante algún tiempo.
- onRestart():

  Se ejecuta cuando una actividad que estaba en estado "*detenido*", va a volver a mostrarse al usuario.
- onDestroy():

  Se ejecuta antes de que la actividad sea destruida.
- onSaveInstanceState(Bundle):

  Este método es ejecutado por "*Android*" para permitir a las actividades guardar el estado de la instancia, como puede ser la posición de un cursor en un campo de texto por ejemplo. La implementación predeterminada para una actividad ya guarda

el estado de todos los controles de la misma por defecto, así que normalmente no será necesario implementar este método.

### 2.1.3.3 Principales componentes del SDK Android

El "SDK Android" consiste en un kit de desarrollo que proporciona "Android" para el desarrollo de aplicaciones para dicha plataforma de manera gratuita.

El "SDK Android" esta compuesta por una serie de objetos que servirán para realizar una aplicación "Android". Los más importantes son los que se mencionan a continuación:

- Actividades ("Activity"):

  Ya hemos hablado antes de ellas, pero en definitiva son las entidades que definen la interfaz de una pantalla de usuario. Una aplicación podrá contener varias y se utilizarán para llevar un control de las diferentes fases de la aplicación.

- Intenciones ("Intent"):

  Son las entidades que permitirán al desarrollador definir el paso de una actividad.

- Servicios ("Service"):

  Son tareas que se ejecutan en segundo plano, con independencia de que el usuario este visualizando nuestra aplicación o no. Como por ejemplo cuando estamos escuchando música, estemos o no visualizando la pantalla de nuestro reproductor, la música debe seguir sonando. En estos casos deberemos utilizar servicios.

- Proveedores de contenido ("ContentProvider"):

  Consiste en un conjunto de datos a los que se tiene acceso mediante una API personalizada. Estos objetos se utilizan cuando una aplicación quiere compartir sus datos de manera global, es decir, con otras aplicaciones.

## 2.2 INTERFACES MULTIMODALES

### 2.2.1 Síntesis de Voz (Text To Speech)

#### 2.2.1.1 Composición

La síntesis de voz consiste en la creación de ondas de sonido artificiales semejantes al habla humana. En inglés se conoce como *Text To Speech* por la conversión de texto a voz.

Los sistemas de síntesis de voz tienen dos partes diferenciadas. La primera parte se llama *front-end*.

Es la parte encargada de recoger el texto y procesarlo sustituyendo abreviaturas o números en sus equivalentes para facilitar la transformación posterior. A menudo se conoce como normalización del texto o pre-procesado. También hace una distinción entre las partes de la oración en base a la prosodia de la frase (si es interrogación, exclamación, frase afirmativa, negativa, etc.). Separa palabras y les asigna la trascripción fonética.

La segunda parte de los sistemas de síntesis de voz se llama *back-end*. Consiste en crear las ondas de sonido mediante las configuraciones que han tenido lugar en la *front-end*.

#### 2.2.1.2 Historia

Mucho antes del desarrollo del procesado de señal moderno, los investigadores de la voz intentaron crear maquinas que produjesen habla humana. El Papa Silvestre II (1003), Alberto Magno (1198-1280) y Roger Bacon (1214-1294) crearon ejemplos tempranos de 'cabezas parlantes'. Autómatas capaces de crear ondas de sonido semejantes a la voz humana.

Por supuesto, estos sonidos eran guturales y casi no se podía entender lo que decían.

En 1779, el científico danés Christian Gottlieb Kratzenstein, que trabajaba en esa época en la Academia Rusa de las Ciencias, construyo modelos del tracto vocal que podían producir las cinco vocales largas (a, e, i, o, u). Wolfgang von Kempelen de Vienna, Austria, describió en su obra menschliche Sprache Mechanismus Beschreibung der sprechenden Maschine ("mecanismo del habla humana con descripción de su maquina parlante", J.B. Degen, Wien) una maquina accionada con un fuelle. Esta maquina tenia, además, modelos de la lengua y los labios, para producir consonantes, así como vocales. En 1837 Charles Wheatstone produjo una 'maquina parlante' basada en el diseño de von Kempelen, y en 1857 M. Faber construyo la maquina 'Euphonia'. El diseño de Wheatstone fue resucitado en 1923 por Paget.

En los años 30, los laboratorios Bell Labs desarrollaron el VOCODER, un analizador y sintetizador del habla operado por teclado que era claramente inteligible. Homer Dudley

refino este dispositivo y creo VODER, que exhibió en la Exposición Universal de Nueva York de 1939.

Los primeros sintetizadores de voz sonaban muy robóticos y eran a menudo inteligibles a duras penas. Sin embargo, la calidad del habla sintetizada ha mejorado en gran medida, y el resultado de los sistemas de síntesis contemporáneos es, en ocasiones, indistinguible del habla humana real.

A pesar del éxito de los sintetizadores puramente electrónicos, sigue investigándose en sintetizadores mecánicos para su uso en robots humanoides. Incluso el mejor sintetizador electrónico esta limitado por la calidad del transductor que produce el sonido, así que en un robot un sintetizador mecánico podría ser capaz de producir un sonido más natural que un altavoz pequeño.

El primer sistema de síntesis computerizado fue creado a final de la década de 1950 y el primer sistema completo texto a voz se finalizo en 1968. Hacia finales de los años 70, aparecieron las primeras maquinas capaces de convertir texto tecleado en voz (conversor texto-voz), que junto con los programas de reconocimiento óptico de caracteres (Optical carácter Recognition en ingles) produjeron los primeros sistemas comerciales para leer libros en voz alta. Uno de los más famosos es la Kurzweil Reading Machine, que por su precio en aquella época, solo estaba accesible en algunas bibliotecas importantes del mundo, en particular la del MIT (Massachussets Institute of Technology). Fue precisamente en esa Universidad donde se desarrollo uno de los primeros conversores texto-voz del mundo (el MITTalk). Este sistema, fue convertido en producto por la Empresa Telesensory Speech Systems (más tarde Speech Plus Inc y después adquirida por Centigram). El producto se llamaba Prose 2000 y convertía en voz todo texto enviado a su puerto serie en formato ASCII. La primera versión funcionó solo para el idioma ingles americano. Otros sistemas le siguieron como el DEC-Talk, el Klat-talk, el Infovox, y muchos otros.

Unos años mas tarde empezaron a aparecer conversores texto-voz en otros idiomas, español, francés, sueco, alemán, italiano... etc.

Posteriormente, ya a finales de los anos 80, las principales operadoras telefónicas del mundo tomaron cartas en el asunto, y produjeron sus propios conversores texto a voz, en un conjunto de idiomas diverso. Cabe citar Bell Labs de ATT, mas tarde escindida en Lucent Technologies y ATT Research, British Telecom., France Telecom., Deutsche Telecom., CSELT, NTT y, por descontado, Telefónica [3].

El interés de todas estas últimas, centrado sobre todo en la automatización de servicios de información telefónica, en los que los datos disponibles están sobre todo almacenados en el ordenador en modo texto.

Precisamente los servicios de información y atención telefónica automática son uno de los pilares económicos importantes de todos los desarrollos actuales de la Tecnología del Habla.

#### 2.2.1.3 Sistemas de Síntesis de Voz

A la hora de crear una onda de voz hay que tener en cuenta dos aspectos fundamentales que tiene que tener el resultado final.

La naturalidad y la inteligibilidad. Los sistemas de síntesis, buscan un equilibrio de ambas características.

La naturalidad es la cualidad por la que una onda sintética se parece a la voz humana (conjunto de factores que afectan la pronunciación de una manera global, como la entonación, el ritmo y la intensidad del habla).

La inteligibilidad es la cualidad de poder entender la onda de voz sintética sin necesidad de esfuerzo.

Un conversor de texto a voz consta de tres bloques [4]:

- Análisis lingüístico del texto:

Como se ha dicho previamente, se realiza un pre-procesado para insertar correctamente los equivalentes tanto de abreviaturas como de números, etc.

También se divide la frase por palabras y se separan las partes gramaticales. Finalmente este bloque también engloba la conversión del texto en los símbolos fonéticos basándose en las reglas gramaticales del idioma.

Dependiendo de la sofisticación del software, pueden haber hasta cuatro niveles de conversión basándose en el nivel fonético (el más sencillo que atiende a la pronunciación de las palabras), en el nivel sintáctico (atendiendo a la estructura gramatical), en el nivel semántico (atendiendo al significado de la frase a procesar) o al nivel pragmático (atendiendo al significado del discurso).

Para mejorar la naturalidad del resultado final, se puede crear una prosodia concreta de cada palabra dependiendo del lugar que ocupa en la frase a procesar.

- Estudio de la frase y su entonación o prosodia

Este bloque central se encarga de estudiar la frase en su conjunto para poder darle una entonación correcta. También se encarga de estudiar los aspectos relacionados con la prosodia como son la entonación, la melodía (ritmo), y la intensidad del texto. Estos aspectos se han ido alcanzando y perfeccionando por medio del ensayo y error hasta conseguir una entonación y melodía adecuados.

Actualmente tienen bases de datos que alcanzan la prosodia correcta de las frases automáticamente por medio de métodos estadísticos sobre una base de datos preestablecida.

Este apartado es importante para generar una voz artificial lo más natural posible, ya que una buena entonación y la melodía ayudan a la calidad final de la onda sintetizada.

La entonación es la evolución de la frecuencia fundamental (*pitch*) a lo largo del texto y la melodía o el ritmo incluye tanto las duraciones de cada uno de los segmentos como de las pausas entre palabras.

En el idioma español, la entonación consta de una rama ascendente que comprende desde el primer sonido hasta el primer acento tónico (rama intensiva) y a partir de aquí se mantiene subiendo y bajando la entonación hasta la parte del último acento (rama distensiva).

La elevación de la rama distensiva significa que la frase no está completa. Si esta rama es de entonación descendente, significa que la frase finaliza (rama conclusiva).

Finalmente si se produce una elevación y un descenso en la misma frase, significa que la frase es interrogativa.

- Síntesis de voz

El último bloque es el sistema de síntesis de voz. En este bloque existen dos tecnologías principales: la síntesis concatenativa que se basa en segmentos grabados de voz humana real y la síntesis por formantes que se basa en crear una onda de sonido partiendo de cero, configurando las frecuencias y las demás características.

Una vez encontradas las unidades mínimas tanto en el de síntesis concatenativa como en el de síntesis por formantes, se aplica un postprocesado segmental.

Este post-proceso tiene la función de asegurar la continuidad de la entonación, la intensidad y la melodía al pasar de una unidad mínima a otra, manteniendo el tono constante sin saltos ni defectos fonéticos.

#### 2.2.1.4 Tipos de Síntesis de Voz

##### 2.2.1.4.1 Concatenativa

Existen tres tipos de síntesis concatenativas:

- Síntesis por selección de unidades:

Este tipo de síntesis concatenativa trae consigo una gran base de datos donde están almacenadas las grabaciones del habla.

La base de datos se compone de unidades que pueden ser fonemas, sílabas, palabras, frases u oraciones. Con estos datos, se modela la onda de voz atendiendo a los datos analizados anteriormente.

El problema de esta tecnología era que para cada prosodia determinada de la frase se tendría que elaborar y parametrizar dicha frase para unir los distintos segmentos. La forma de unir los segmentos se revoluciono a partir de la aparición del procesamiento segmental llamado PSOLA (Pitch-Synchronous Over-Lap and Add).

Permite modificar la frecuencia fundamental de una señal de un modo sencillo, actuando directamente sobre la representación de la misma en el dominio del tiempo, pudiendo utilizarse por tanto para formar la curva prosodica de los mensajes a emitir.

Consiste en la separación de las distintas unidades, duplicando su periodo al doble y superponiendo las unidades para que no tengan saltos cuantitativos a la hora de la unión. Se utiliza para aumentar la naturalidad de la voz sintética.

La síntesis por selección de unidades de propósito general (sin un tema específico) necesita una base de datos extensa para tener una buena calidad y naturalidad.

Es posible crear una voz natural e inteligible con una base de datos pequeña si es con un propósito concreto sobre un determinado tema, como por ejemplo, el sistema de aviso de una terminal de autobuses que esta determinado a decir lo mismo una y otra vez sin salirse del guión establecido.

- Síntesis por difonos:

Utiliza una base de datos mas compacta.

Contiene todos los difonos del lenguaje. A partir de estos difonos y atendiendo a los parámetros obtenidos en los procesos anteriores, se realiza una búsqueda estadística, seleccionando el difono que mas se ajuste.

Como los difonos ya contienen una característica prosodica del lenguaje, no es necesario realizar el proceso segmental de la unión por unidades pero si conviene hacer un post-procesado para suavizar la unión de los mismos.

El resultado final es una voz robótica poco natural y parcialmente inteligible. Por este motivo no se usa en aplicaciones comerciales pero si en investigación ya que tiene programas de libre distribución.

- Síntesis especifica para un dominio:

En esta síntesis, se tiene una base de datos con palabras y oraciones grabadas directamente de la voz humana. Se utiliza con fines sobre temas muy concretos, como por ejemplo un sistema de sonido que de las horas.

En este sistema se nota mucho el cambio de entonación e intensidad entre palabras. En cambio es uno de los sistemas más naturales que existen puesto que la prosodia y entonación de las palabras como unidades son las de las grabaciones originales. No es valido para oraciones de propósito general (para cualquier tema) debido a la mínima base de datos.

#### 2.2.1.4.2 Por Formantes

Esta tecnología crea zonas de concentración de energía en el espectro del sonido sintetizado, lo que imita el sonido de la voz.

La síntesis de formantes no usa muestras de habla humana en tiempo de ejecución. En lugar de eso, la salida se crea usando un modelo acústico. Parámetros como la frecuencia fundamental y los niveles de ruido se varían durante el tiempo para crear una forma de onda o habla artificial.

Se basa en la utilización de filtros para crear los distintos efectos sobre las ondas que provocarían los aparatos fonadores humanos.

Estos filtros modifican los siguientes parámetros acústicos como muestra la siguiente figura.

| Parámetro | Base fisiológica | Acústica |
|---|---|---|
| AV amplitud de la sonoridad | Amplitud de la vibración de las cuerdas vocales | Intensidad de la onda sonora |
| Fofrecuencia fundamental | Frecuencia de vibración de las cuerdas vocales | Frecuencia fundamental |
| F1 primer formante | Primera resonancia del tracto vocal | Frecuencia del primer formante |
| F2 segundo formante, F3 tercer formante... | Segunda resonancia del tracto vocal, tercera resonancia del tracto vocal... | Frecuencia del segundo formante, frecuencia del tercer formante... |
| B1 amplitud de banda del primer formante, B2, B3... | Configuración del tracto vocal | Ampitud de banda del primer formante... |
| AK Amplitud del ruido de fricción | Tipo de constricción | Intensidad de los componentes aperiódicos |
| K1, K2 Frecuencia del ruido de fricción | Tipo de constricción | Frecuencia de los componentes aperiódicos |
| AH Amplitud de la aspiración | Fricción en el tracto vocal | Componentes aperiódicos de los formantes |
| N1 Frecuencia del formante nasal | Resonancia del tracto nasal | Frecuencia del formante nasal |
| AN Amplitud de la nasalidad | Resonancia del tracto nasal | Intensidad de la resonancia nasal |

**Figura 2-7: Parámetros de la Síntesis de Voz**

### 2.2.2 Reconocimiento de Voz

El reconocimiento del habla o voz, es un sistema capaz de transcribir un mensaje oral en texto o emitiendo órdenes acordes, independientemente del hablante. Se basa en la comparación con un modelo acústico recogido en una base de datos.

Su implantación comercial nos ha llegado en forma de compañías telefónicas, acceso a discapacitados visuales, etc.

#### 2.2.2.1 Historia

En la época de 1870, Alexander Graham Bell invento el teléfono gracias a querer implementar un sistema capaz de ayudar a las personas con discapacidades físicas auditivas.

AT&T Bell Laboratories, en los anos 1910 inventan un sistema electrónico capaz de reconocer los 10 primeros números en ingles con un índice de acierto del 99%. Este sistema necesitaba de una gran configuración interna para cada locutor.

A mediados de los anos 60 los sistemas empiezan a incorporar técnicas de normalización del tiempo (que dos palabras iguales dichas más o menos rápidas pudieran reconocerse igual), se centran en locutores individuales y con un vocabulario muy reducido (alrededor de 50 palabras).

Estos sistemas tenían en común el segmentado del habla, es decir, las palabras se pronunciaban con cierto espacio entre si para que el sistema pudiera reconocerlas.

A principios 1970 se produce el primer producto comercial de reconocimiento de voz, el VIP100 de Threshold Technology Inc. que utiliza un vocabulario pequeño, dependiente del locutor, y reconocía palabras discretas [5].

#### 2.2.2.2 Tipos de Reconocimiento del Habla

Existen dos tipos de reconocimiento automático del habla: DirectVoice Input (DVI), que se basa en el reconocimiento de unos cientos de palabras para trasladar esa información a otra aplicación, y el otro es el Large vocabulary continuous speech recognition (LVCSR) que se usa en la creación de documentos pudiendo reconocer frases extensas [6].

Ambos sistemas se basan en la comparación de fonemas.

En la base de datos se almacenan los fonemas que componen el lenguaje y lo comparan con la palabra o frase que el locutor emite.

Esta comparación es compleja ya que utilizan unas técnicas basadas en modelos ocultos de Markov (Hidden Markov Models – HMM [7]).

Su característica principal es que está compuesto de estados y a estos estados se llega o se cambia dependiendo de la probabilidad de unos parámetros de entrada.

La calidad de un sistema de reconocimiento del habla se mide en el porcentaje de error y velocidad.

El error se da en tanto por ciento de la frase a procesar. Así si se esta procesando una frase con 10 palabras, tendremos un error en función de las palabras reconocidas correctamente, de las que están cambiadas de sitio, de las que no están y de las que aparecen pero no se han pronunciado.

Este error es comúnmente conocido como Word Error Rate (%) (WER). La velocidad es un factor a tener en cuenta en los sistemas de tiempo real.

En estos sistemas cumplir los plazos es esencial, por ello el software asociado al reconocimiento de voz tiene que ser lo mas rápido posible.

Para poder procesar la onda de sonido, se divide periodos de 20 ms y se analizan los parámetros en el dominio del tiempo tales como frecuencia, amplitud, silencios y armónicos y otros parámetros en dominio discreto (transformadas de Fourier).

Se hace un primer barrido de esos 20 ms y se detectan los silencios para poder separar las palabras. En la base de datos se almacenan patrones de comparación de los distintos fonemas del lenguaje.

Con los datos obtenidos y por medio de un sistema estadístico complejo se obtienen unos resultados aproximados.

Algunos programas muy sofisticados llegan a hacer varios barridos de este tipo para aumentar el porcentaje de aciertos.

Estos modelos estadísticos son muy diversos pero los dos predominantes son las cadenas ocultas de Markov (HMM) y las redes neuronales

El primero se basa en la comparación de una secuencia de patrones generada por medio de un sistema de estados. Esta secuencia de estados tiene la característica que no se puede observar o determinar directamente.

Los estados pueden ser Left-to-right (de izquierda a derecha) o pueden ser ergodicos (se puede llegar desde cualquier estado). En la Fig. 2-8 [8] se detallan los tipos de estados de los modelos ocultos de Markov.

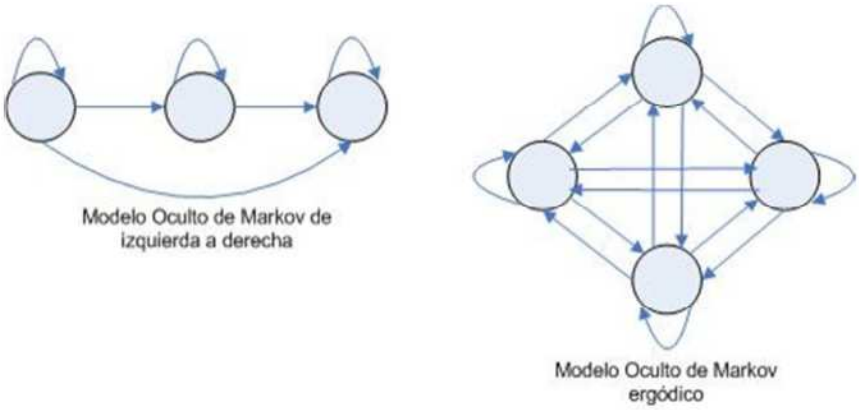


**Figura 2-8: Modelos Ocultos de Markov**

En ambos casos, se modifica el estado por medio de las probabilidades estadísticas de unos parámetros en las ondas procesadas.

Las redes neuronales se basan en unos objetos de programación que reciben entradas (parámetros de las transformadas de Fourier) y dependiendo del peso y del algoritmo interno de estos objetos darán una señal de salida para poder hacer conexiones con otros objetos formando una red.

Estos objetos tienen un umbral de comparación para generar una salida acorde a los patrones a comparar.

El objetivo es el mismo que el del sistema de los modelos ocultos de Markov: comparar los patrones de la onda de sonido con otros patrones almacenados para decidir si es o no la palabra o frase correcta que se quiere procesar.

Cada estado lleva asociado unos determinados argumentos que en conjunto dan como resultado el texto de la frase procesada.

### 2.2.3 Elección del SKD de Síntesis y Reconocimiento de Voz

El API de Android ofrece por defecto tanto síntesis como reconocimiento de voz, a través de los paquetes ““android.speech”, con lo que es el elegido para el proyecto. Más información en el capítulo 5.2.2 y 5.2.3

## 2.3 REALIDAD AUMENTADA

### 2.3.1 Introducción a la Realidad Aumentada

Existen muchas definiciones de Realidad Aumentada (RA), pero hay dos que son las más comúnmente aceptadas Ronald Azuma [9] y Paul Milgram [10].

La definición de Azuma dice que la RA:

- Combina elementos reales y virtuales.
- Es interactiva en tiempo real.
- Está registrada en 3D.

Se dice que la RA es un híbrido entre el mundo real y el mundo virtual. Paul Milgram clasificó por primera vez los distintos espacios de realidad "mixta" desde el punto de vista de continuidad del contexto. En la siguiente figura, Milgram definía un modelo "continuo virtual". Este concepto describe que existe una escala continua entre lo completamente real y lo completamente virtual. Entre ambos existe la virtualidad aumentada (está más próxima al entorno virtual) y la RA (más próxima al entorno real). La posición de la RA en este sencillo esquema indica que, en esta, la mayor parte de la información es real, incluyéndose complementos virtuales.

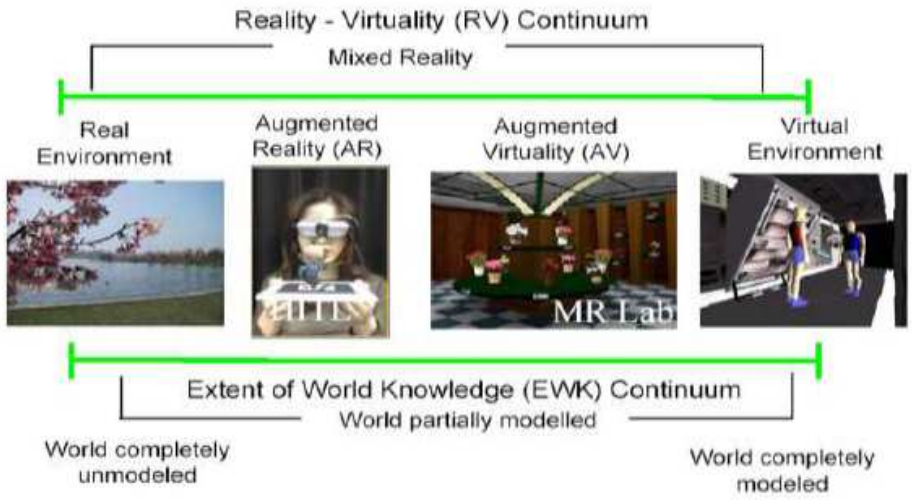


**Figura 2-9: Reality-Virtuality Continuum**

Estos complementos virtuales suelen ser modelos creados por ordenador, objetos 3D, o 2D aunque son menos frecuentes. Estos objetos podrían ser incluso texto, iconos o cualquier imagen que amplíe la información de la imagen real. En algunas aplicaciones específicas se

ha utilizado vídeo, sonido e incluso olores, pero de aquí en adelante sólo haremos referencia a los objetos 3D como parte de la RA.

Actualmente, el término RA se ha extendido enormemente debido al creciente interés por esta tecnología del público en general. Sin embargo en términos generales podemos decir que la RA permite enriquecer la perspectiva del usuario mediante la superposición de objetos virtuales en el mundo real de manera que convenza al usuario de que estos objetos forman parte del entorno real o ampliar su información.

Se puede decir que la RA adquiere presencia en el mundo científico a principio de los años 1990, cuando el avance de la tecnología permitió combinar imágenes generadas por ordenador con la visión que tiene el usuario del mundo real, es decir, cuando aparecieron los primeros ordenadores de procesamiento rápido, las técnicas de renderizado de gráficos en tiempo real, y los sistemas de tracking portátiles de precisión.

Hoy en día la RA está siendo perfeccionada por diversos grupos e investigación de todo el mundo en las tecnologías involucradas, tales como el tracking de la posición del usuario, procesado de la señal, visualización de la información, visión artificial, generación de imágenes virtuales, renderizado de gráficos, estructuración de la información, y hasta computación distribuida.

La RA se ha aplicado con éxito a numerosas áreas. Entre ellas cabe citar: publicidad y marketing, turismo, entretenimiento, cirugía (Figura 2-10), industria (Figura 2-10), tratamiento de fobias (Figura 2-11) y aprendizaje (Figura 2-12). Estos son sólo algunos ejemplos, pero la RA se va introduciendo cada vez más en las actividades cotidianas de cualquier persona.

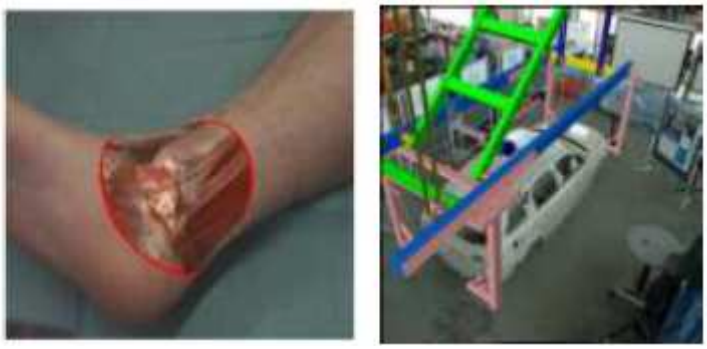

**Figura 2-10: Ejemplos de RA Aplicada a la Cirugía y a la Industria**

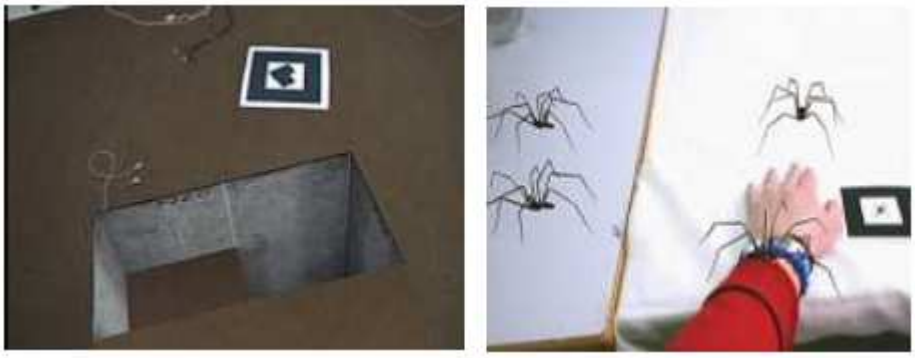

**Figura 2-11: Ejemplo de RA aplicada al tratamiento de fobias**

**Figura 2-12: Ejemplo de RA aplicada al aprendizaje**

En todo sistema de RA son necesarias cuatro tareas principales para poder llevar a cabo el "aumento" de la realidad. Éstas son:


**Figura 2-13: Etapas del proceso de Realidad Aumentada**

Siguiendo el gráfico de la anterior figura, en primer lugar, se captura la escena real con la cámara. Después se procede a su procesamiento con el fin de solucionar uno de los principales problemas de la RA: el seguimiento del punto de vista (Viewpoint tracking). Éste es un problema clave dado que condiciona el posicionamiento de los objetos virtuales para su visualización por parte del usuario.

Posteriormente, se pasa a la fase de renderizado del objeto virtual, que debe estar colocado en la posición anteriormente calculada. Y finalmente se pasa a la etapa de visualización, dónde se superponen las capas real y virtual. En algunas aplicaciones de RA puede existir

una fase intermedia entre la tercera y la cuarta que es la que comprende las interacciones del usuario.

Una de las principales características que diferencia a unas aplicaciones de RA de otras es precisamente el método utilizado para el Viewpoint tracking. Para algunas aplicaciones que requieren mucha precisión, como es el caso de la cirugía, el correcto posicionamiento del objeto virtual es esencial.

Existen básicamente dos métodos totalmente diferentes para solucionar el problema del viewpoint tracking:

- **Localización espacial utilizando GPS**: en base a las coordenadas GPS, el software calcula la posición de los objetos virtuales a añadir a la escena. Aunque se dice que la tecnología GPS puede llegar a tener precisión de hasta centímetros, lo habitual son unos pocos metros de precisión, debidos al error de triangulación de los satélites. Este método por tanto, no puede utilizarse en sistemas de RA donde la precisión sea crítica. Por el contrario, éste es el método más rápido debido a que sólo necesita conocer la posición del GPS, sin tener que pasar por fase de reconocimiento que suele tener un coste computacional más alto. Se puede aumentar la precisión de estos sistemas mediante brújulas digitales (para conocer la dirección hacia la que estamos observando) u acelerómetros (para conocer la orientación de la cámara). Generalmente, los nuevos smartphones ya integran los tres sensores.

- **Reconocimiento espacial utilizando visión artificial**: éste es un método bastante más complejo a nivel de software y puede, a su vez, dividirse en otros dos:
    - **Reconocimiento de marcadores físicos (o marker tracking**): este método se basa en la detección de lo que se conoce como “markers”. Suele consistir en un cuadrado blanco y negro con un patrón asimétrico en el interior.

**Figura 2-14: Ejemplo de marcador de RA**

    - **Reconocimiento espacial sin marcadores (o markerless tracking)**: ésta es una técnica aún más compleja. En este caso el software de RA debe ser capaz de reconocer diferentes objetos que componen la escena del mundo real. Este método integra muchas técnicas de visión artificial. En las aplicaciones actuales que utilizan este método se suele restringir el reconocimiento a ciertos objetos como caras o manos humanas, superficies, u objetos con una forma concreta. Estos objetos se denominan *targets* [11].

De aquí en adelante nos centraremos únicamente en los sistemas de RA pertenecientes al segundo grupo, es decir, sistemas con reconocimiento espacial basado en visión artificial.

En la siguiente imagen puede verse un ejemplo de la ejecución completa del proceso de RA con markers.

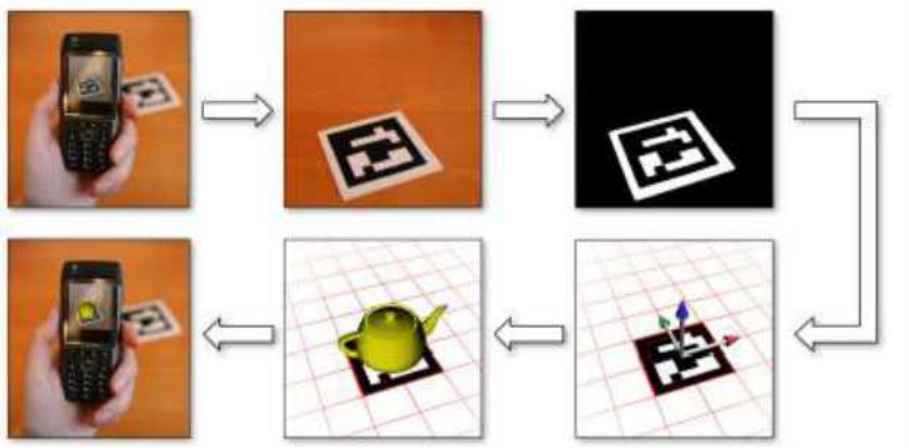

**Figura 2-15: Diagrama de Funcionamiento ARToolkitPlus**

### 2.3.2 Realidad Aumentada en Dispositivos Móviles

Como ya se ha comentado, la investigación y desarrollo de software para el uso de la RA está en continuo auge. Sus múltiples aplicaciones y su atractivo aspecto visual están haciendo que cada vez más usuarios se interesen por ellos, y que los desarrolladores de software lo integren en sus proyectos.

A continuación se ofrece una visión general de la trayectoria histórica del desarrollo de aplicaciones de RA hasta la introducción en el mundo de los dispositivos móviles. Esta revisión es un resumen del estudio de Daniel Wagner [12].

**1968**

Ivan Sutherland crea el primer sistema de RA, que también es considerado como el primer sistema de realidad virtual. Se utiliza un HDM seguido por dos mecanismos de rastreo 6DOF, un tracker mecánico y un dispositivo ultrasónico. Debido a la baja capacidad de proceso de los ordenadores del momento, sólo se podían mostrar gráficos alámbricos en tiempo real.

**Figura 2-16: Sistema de RA**

**1982**

Aparece el primer ordenador portátil con diseño plegable, el Grid Compass 1100. Tenía un procesador Intel 8086, 350 Kbytes de memoria y una pantalla con 320x240 píxeles de resolución; lo cuál era extraordinario para esa época. Sin embargo sus 5kg de peso lo hacían difícilmente portátil.

**1992**

Tom Caudell y David Mizell acuñan el término "realidad aumentada" para referirse a la superposición del mundo real con información generada por ordenador. Se discuten las ventajas de la RA frente a la realidad virtual, como la menor potencia de proceso requerida debido al menor peso de las imágenes a renderizar. También reconocen el aumento de los requerimientos de registro con el fin de alinear mundo real y virtual.

IBM desarrolla el primer smartphone que saldría al mercado en 1993. El teléfono tenía 1MB de memoria y una pantalla táctil en blanco y negro de 160x293 píxeles de resolución.

**1994**

Milgram y Kishino describen en su "Taxonomía de la Realidad Mixta" el conocido término del continuo de Milgram (Reality-Virtuality Continuum).

**1995**

Rekimoto y Katashi crean NaviCam, una estación de trabajo con cámara montada que se utilizaba para el seguimiento óptico. El equipo detectaba los marcadores codificados en la imagen de la cámara en vivo y mostraban información directamente sobre la secuencia de vídeo.

**1996**

Rekimoto presenta uno de los primeros sistemas de marcadores para permitir el seguimiento de la cámara con seis grados de libertad, los marcadores de matriz 2D (cuadrados con forma de código de barras, ver figura mostrada a continuación).

**Figura 2-17: Primer Marcador 2D que permite 6DOF**

**1997**

Ronald Azuma presenta el primer estudio sobre RA. En él presenta su definición de RA, hoy en día ampliamente reconocida.

**1998**

Bruce Thomas presenta “Map in the hat”, un ordenador montado en una mochila que incluía GPS, brújula electrónica y HMD.

**Figura 2-18: Map in the Hat the B.H. Thomas**

**1999**

Kato y Billinghurst presentan ARToolKit, una librería de seguimiento con 6DOF, utilizando markers para el reconocimiento de patrones. ARToolKit está disponible como código abierto bajo la licencia GPL y es todavía muy popular en la comunidad RA.

Hollerer et al. presentan el primer sistema de RA móvil basado en GPS y sensores inerciales.

Nace el teléfono Benefon Esc! NT2002, el primer teléfono móvil GSM con GPS integrado. Debido a la limitación de memoria el teléfono descargaba los mapas de navegación bajo demanda.

Se define el protocolo Wireless Network, comúnmente conocido como WIFI.

**2000**

Julier et al. presentan BARS (Battlefield Augmented Reality system). El sistema consiste en un sistema portátil con conexión wifi y HMD. El sistema muestra de forma virtual una escena de batalla con información adicional sobre la infraestructura del entorno y sobre posibles enemigos.

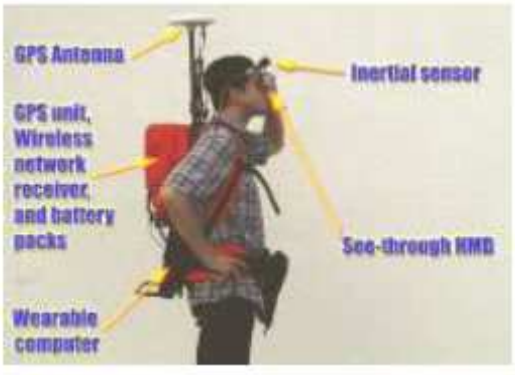


**Figura 2-19: Sistema de RA portátil**

La empres Sharp lanza el primer teléfono móvil comercial con cámara integrada, el JSH04. La resolución de la cámara era de 0.1 Megapixels.

**2001**

Fruend et al. presentan AR-PDA, un prototipo para construir sistemas de RA sobre PDA's. El diseño inicial incluye el aumento de imágenes reales con objetos virtuales, para por ejemplo ilustrar el funcionamiento de algunos electrodomésticos y su interacción con ellos.

**2002**

Kalkusch et al. presenta una aplicación de RA para guiar al usuario en el interior de un edificio hacia su destino. El sistema superpone un modelo de la estructura del edificio según se va avanzando por él, todo ello sobre un HDM. Utiliza marcadores de ARToolKit.

**Figura 2-20: Sistema de guiado basado en RA**

**2003**

Wagner y Schmalsteig crean un sistema de RA de guiado en interiores sobre una PDA. La aplicación provee al usuario mediante objetos aumentados de la información para llegar a su destino. Se trata del primer sistema autónomo e independiente. Utiliza Windows Mobile y está implementado con ARToolKit.

**2004**

Mohring et al. presentan un sistema para el posicionamiento con marcadores 3D en teléfonos móviles.

Rohs y Gfeller presentan Visual Codes, un sistema de marcadores 2D para teléfonos móviles. Estos marcadores pueden utilizarse sobre objetos físicos para superponer información virtual sobre dicho objeto.

**2005**

Henrysson consigue portar ARToolKit para poder ejecutarlo en el sistema operativo Symbian. Basado en esta tecnología, presenta AR-Tennis, la primera aplicación de RA colaborativa para teléfonos móviles.

Reitmayr et al. presentan un modelo híbrido de sistema de RA de seguimiento en entornos urbanos. El sistema captura en tiempo real la imagen del entorno con una PDA, utilizando la misma para la visualización de la escena. El sistema combina diferentes sistemas de posicionamiento para aumentar la precisión de la localización.

**2008**

Wagner et al. presentan el primer sistema de RA 6DOF con tracking de marcadores naturales para dispositivos móviles, consiguiendo tasas de hasta 20 frames por segundo (fps). El sistema modifica los conocidos métodos SIFT y Ferns para mejorar la velocidad, optimizar y reducir la memoria necesitada. METAIO presenta en el ISMAR'08 un sistema comercial de RA para el guiado en un museo utilizando marcadores naturales.

Mobilizy lanza Wikitude, una aplicación que combina el GPS y la brújula digital para mostrar datos de la wikipedia sobre lugares u objetos en sistemas Android.

**2009**

Kimberly Spreen et al. desarrollan ARhrrr!, el primer videojuego de RA con una calidad gráfica al nivel de los juegos comerciales. Esta aplicación utiliza el kit de desarrollo Tegra de Nvidia, optimizado para las GPU's del momento. Todo el procesamiento se realiza en la GPU, salvo el referido al posicionamiento, haciendo que la aplicación funcione con un alto ratio de frames por segundo.

A día de hoy están al alcance de casi cualquier persona smartphones con las prestaciones necesarias para servir como soporte de estas aplicaciones y sacar el máximo provecho de ellas. La mayor parte de los dispositivos móviles adquiridos por los usuarios en el último año disponen de cámara de vídeo de alta resolución y alta capacidad de procesado. Este hecho, y los recientes estudios de mercado, hacen pensar que la integración de la RA en el creciente mercado de los dispositivos móviles forma un campo de investigación de interés [13].

Como se ha visto en el resumen de antecedentes, Wagner y Schalstieg fueron los primeros en identificar el potencial de los dispositivos móviles para RA. En la comparativa que muestra la siguiente tabla se observa cómo han mejorado los dispositivos móviles en los aspectos más relevantes para RA: procesadores más rápidos, más memoria, mejores interfaces de entrada, pantallas más grandes y de mayor calidad gráfica, más sensores y mejora de las posibilidades de conexión.

| | 2003 | 2012 |
|---|---|---|
| CPU | 400 MHz Intel xScale | Doble núcleo ARM 1.2 GHz |
| RAM | 64 MB | 1GB |
| Soporte Hardware GFX | ninguno | OpenGL ES |
| Cámara | Cámara color 320x240 frontal y trasera integrada por CP jacket | Cámara frontal y trasera de 2 a 8 Megapixels |
| Display | 3.8", 16-bit 240x320 | 4,3", 800x480 |
| Interfaz | Huella dactilar | Pantalla táctil y teclado |
| Sensores | Wifi<br>Bluetooth<br>GPRS | GPS, magnetómetro, RFID, acelerómetro, giroscopio de 3 ejes, sensor de proximidad, sensor de luz ambiente |
| Conexión | 900-1200 mAh | Wifi<br>Bluethooth<br>USB<br>3G, GPRS |
| Batería | 900-1200 mAh | 1650 mAh |
| Sistema Operativo | Windows Pocket PC<br>Windows Mobile | Android<br>iOs |
| Precio | 900€ | 400-500€ |

**Figura 2-21: Comparación de hardware de dispositivos móviles para RA, de 2003 a 2012**

Con la llegada de los sistemas operativos iPhone OS y Android, el término Smartphone se acuñó para indicar la capacidad de estos dispositivos para la implementación de aplicaciones informáticas complejas. La mejora de interfaces, y en particular la introducción de la pantalla táctil, y el fácil acceso a las aplicaciones, "Apps", por parte de los usuarios, explica su gran éxito comercial. Este fuerte interés comercial también ha traído consigo la mejora de las herramientas de desarrollo de software dedicado a smartphones.

En definitiva, en conjunto, el nuevo paradigma de dispositivos móviles ofrece todos los ingredientes necesarios para convertir la RA, de un software de uso casi exclusivo a su uso por un público masivo.

Sin embargo, a pesar de todas las mejoras en aspectos logísticos y técnicos, no cabe duda de que todavía existen importantes obstáculos para una implementación a gran escala de las aplicaciones de RA.

### 2.3.3 Herramientas para la Implementación de Aplicaciones de Realidad Aumentada

A continuación se realiza una revisión de las herramientas para desarrollar aplicaciones de RA con dispositivos móviles más conocidas y gratuitas.

#### 2.3.3.1 ARtoolKitPlus

ARToolKitPlus en una variante de ARToolKit [14], optimizada para el desarrollo de aplicaciones móviles. Fue desarrollada por la Universidad de Graz en 2007. Inicialmente era de código cerrado por lo que no está muy documentada. La librería implementa módulos para el cálculo de la orientación y posición de la cámara relativa a los marcadores en tiempo real. Debido a su elevada demanda más tarde se liberó su código. Sin embargo su poca documentación la hace poco recomendable para el uso por parte de desarrolladores poco experimentados. Además el proyecto ha sido abandonado y la librería ha sido reemplazada por StudierStube Tracker y Studierstube ES desarrollado por la misma universidad.

#### 2.3.3.2 Studierstube ES

StudierStube ES [12] es una extensión del framework de desarrollo Studierstube para dispositivos móviles. Se trata de una librería de visión artificial para la detección de la orientación y posición de marcadores 2D con respecto a la cámara. Fue desarrollada por la Universidad de Graz al igual que su predecesora ARToolKitPlus. Como mejoras de esta destaca el hecho de ser multiplataforma (Windows XP, WindowsCE & Windows Mobile, Symbian, Linux, MacOS, Iphone).

Acepta diferentes marcadores, diferentes a los clásicos utilizados por ARToolKit, pero no reconocer marcas naturales. Se trata de código cerrado (no se distribuye).

Este framework se caracteriza por su gran rendimiento en dispositivos con bajas prestaciones y buen aprovechamiento de la memoria. Permite el uso de hasta 4096 marcadores.

El framework ofrece soporte para la calibración de la cámara mediante el toolbox de Matlab e incluye un algoritmo de umbralización adaptativo para los casos de iluminación variable.

#### 2.3.3.3 Layar

Layar es un navegador de RA, desarrollado para Android e iOS. Tiene licencia privativa por lo que no dispone de acceso al código fuente.

El funcionamiento del software se basa en el geoposicionamiento y no en el reconocimiento de marcas.

Está basado en un sistema de capas que funcionan sobre el navegador de realidad y que pueden ser mostradas o no a elección del usuario. El desarrollador implementa estas capas,

en 2D o 3D, para añadir información aumentada a la imagen real (ver imagen mostrada debajo). El sistema se compone de la aplicación cliente que se ejecuta en el dispositivo, un servidor central que provee los datos y un servidor privado para que el desarrollador gestione esos datos y los envíe al servidor central para finalmente visualizarlos en la aplicación. Las capas definidas por el usuario pueden ser puestas a disposición de la comunidad de manera centralizada. La etapa de renderizado de objetos 3D está optimizado para su uso en dispositivos móviles.

**Figura 2-22: Ejemplo de aplicación realizada con Layar**

Este framework está enfocado especialmente al desarrollo de aplicaciones de turismo y entretenimiento. Utilizando los sensores inerciales integrados en los dispositivos móviles analiza la posición del usuario y le ofrece información de los puntos de interés cercanos a él (museos, monumentos, restaurantes, etc).

### 2.3.3.4 Mixare

Mixare [15] (mix Augmented Reality Engine) es un framework de código abierto para RA, publicada bajo la licencia GPLv3 3. Mixare está disponible para sistemas Android y para iPhone.

Este framework permite construir aplicaciones completas y proporciona funciones para asociar coordenadas espaciales y texto. Es decir, su funcionalidad se resume a permitir asociar texto a localizaciones mediante posicionamiento GPS y acceso a datos por conexión de red. Las visualizaciones de Mixare están limitadas a cajas de texto e imágenes 2D.

### 2.3.3.5 AndAR

AndAR es un SDK de código abierto para el desarrollo de aplicaciones de RA para Android basadas en el reconocimiento de marcadores [16]. Utiliza marcadores del tipo ARToolKit. Permite la carga de objetos 3D con formato .obj. Esta librería se presenta con más detalle en el siguiente apartado.

#### 2.3.3.6 NyARToolkit

NyARToolkit es un SDK de código abierto para el desarrollo de aplicaciones de RA basadas en el reconocimiento de marcadores [17]. Se trata de un framework multiplataforma disponible para Android, Java, C#, AS3, C++ y Processing. Utiliza marcadores del tipo ARToolKit, y dispone de soporte para diferentes formatos 3D (.mqo, .md2, .obj) mediante el uso de la librería min3D. Esta librería se presenta con más detalle en el siguiente apartado.

#### 2.3.3.7 Vuforia

Vuforia [18] es una plataforma de desarrollo de aplicaciones de RA para Android e iOS desarrollada por el departamento de I+D de la empresa Qualcomm en Austria. Esta plataforma fue publicada en 2010 y por ejemplo en el último año se ha desarrollado más de 1.000 aplicaciones con ella. Grandes marcas comerciales han utilizado esta plataforma para las campañas publicitarias de sus productos. Una de las principales ventajas de esta plataforma es que se basa en el reconocimiento de marcas naturales, incluyendo objetos 3D, y que existe una extensión para Unity 3D que permite crear escena virtuales con animaciones y muy completas.

La plataforma se presenta en código abierto aunque su uso con Unity 3D requiere de la adquisición de la licencia de esta. Este SDK es presentado con más detalle en el apartado siguiente.

#### 2.3.3.8 Metaio

Metaio Mobile SDK [19] es una plataforma de desarrollo de aplicaciones de RA para dispositivos Android e iOS creada por la empresa Metaio en Alemania. La empresa dispone de más de 10 años de experiencia en el desarrollo de esta tecnología y posee otras plataformas de desarrollo para PC y Web. Las aplicaciones se basan en el reconocimiento de marcas naturales, e integra la gravedad en los módulos de reconocimiento para añadir precisión.

El código del SDK para móviles ha sido liberado recientemente. Este incluye un motor de renderizado que soporta distintos formatos 3D (.md2 animado y .obj estático). También puede utilizarse con Unity 3D aunque requiere adquirir su licencia. Este SDK es presentado con más detalle en el siguiente apartado.

#### 2.3.3.9 ArUco

ArUCo [20] es una librería para Android que permite desarrollar aplicaciones de realidad aumentada basada en OpenCv y desarrollada por el grupo de investigación "Aplicaciones de la Visión Artificial" (A.V.A) de la Universidad de Córdoba. Permite la utilización de marcadores, tableros de marcadores e interacción con objetos 3d. Este SDK es presentado con más detalle en el siguiente apartado.

## 2.3.4 SDKs para Realidad Aumentada mediante Marcadores

Tras un listado inicial de las plataformas de desarrollo disponibles para el desarrollo de aplicaciones de RA nos centramos a continuación en aquellas de código abierto, especialmente preparadas para su uso con Android y basadas en visión artificial para el posicionamiento de los objetos, más concretamente marcadores.

### 2.3.4.1 AndAR

AndAR es una librería creada en 2010, por Tobias Domhan, especialmente para dispositivos Android mediante una API escrita en Java. Está basada en el proyecto ARtoolKit. Hereda la licencia dual de ARToolKit, por lo que, aunque está licenciado bajo GNU GPL v3, existe la posibilidad de utilizarla en aplicaciones de código cerrado mediante pago de una licencia a ARToolworks.

Esta librería funciona con marcadores básicos. La librería maneja una plantilla de texto para gestionar los marcadores. Estas plantillas pueden generarse mediante el software mk_patt de ARToolKit, que sirve para convertir una imagen patrón en una plantilla que pueda ser reconocida por la librería. En la documentación de ARToolKit se puede encontrar la plantilla para crear nuestro propio marcador, puesto que éste debe tener unas proporciones exactas para poder ser reconocido por la librería: La siguiente figura muestra un ejemplo de marker.

**Figura 2-23: Plantilla marker Artoolkit**

La siguiente figura muestra el diagrama de clases de una aplicación simple que utilice AndAR. A continuación se explica brevemente esta estructura.

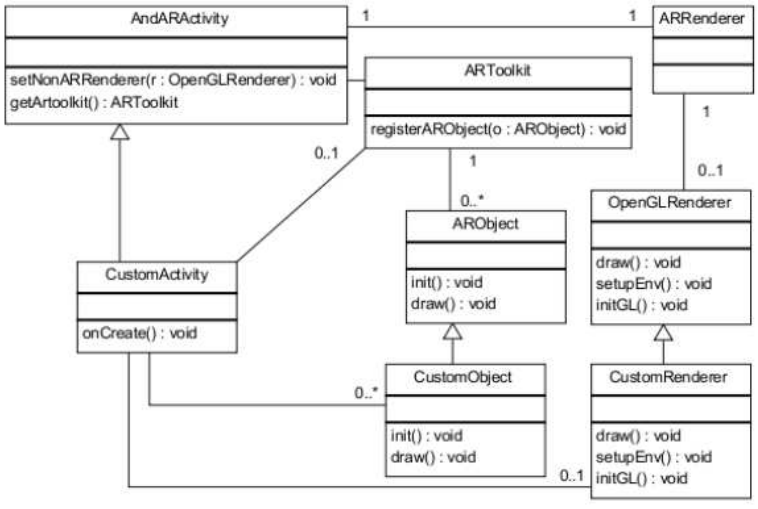


**Figura 2-24: Diagrama de Entidad-Relación de AndAR**

En primer lugar vemos la “actividad prinicipal” AndARActivity. Siempre se creará al menos una actividad extra CustomActivity, que hereda de esta clase abstracta. Esta clase implementa todas las tareas relativas al manejo de la cámara, detección de marcadores y visualización de vídeo. En nuestra clase podremos gestionar las tareas relativas a inserción de objetos, desde onCreate.

Por otro lado, la clase ARRenderer es la responsable de todo lo relacionado con OpenGL. Esta clase se utiliza para implementar el renderizado, y es la que permite mezclar la realidad con los objetos aumentados. Desde esta clase podemos gestionar parámetros de iluminación, posición y cualquier tarea relacionada con la inicialización de OpenGL. En nuestro caso, como se explicará más adelante, se ha extendido la clase ARObjects creando una clase capaz de mostrar modelos en formato .obj.

ARObject es la clase responsable de proporcionar constantemente la información sobre el marcador. Como ya hemos comentado que esta librería está basada en ARToolKit, utiliza los mismos patrones de datos generados para ARToolKit básico, y que pueden crearse mediante la herramienta mk_patt. Ésta es la primera limitación identificada. Esta limitación se basa en que el patrón debe ser almacenado en un archivo dentro de la carpeta assets del proyecto. Esto es imprescindible para poder encontrarlo cuando se crea el objeto.

#### 2.3.4.2 NyARToolkit

NyARToolkit es una librería también basada en ARToolKit creada por Ryo Iizuka en 2008. En este caso no fue desarrollada específicamente para desarrollo aplicaciones Android, sino que el proyecto NyARToolkit es compatible con plataformas Java, C#, ActionScript 3, Silverlight 4, C++, Processing y obviamente Android, aunque no todos al mismo nivel en términos de estabilidad. De esta misma librería surgió otra, FLARToolKit, aún no disponible para Android.

NyARToolkit para Android ha sido desarrollada por un grupo de usuarios de Android en Japón y no existe apenas documentación online, y la que hay está escrita en japonés. Por eso, para entender esta librería hay que sumergirse directamente en el código fuente de Java para Android. Incluso la documentación del código está hecha en japonés.

Esta librería se encuentra aún en fase de desarrollo, pero no tiene una comunidad grande de desarrolladores detrás, por lo que los avances son muy lentos. Hasta hace poco, NyarToolkit para Android dependía del formato MQO (metasequoia) para modelos 3D. Se trata de una aplicación gratuita (de código cerrado) y shareware para modelado y texturas 3D muy popular en Japón, pero en ningún sitio más. Al menos existe un exportador para Blender a formato MQO. Sin embargo, la última versión de NyARToolkit puede importar formatos MD2 y OBJ, ambos también compatibles con Blender.

La estructura de una aplicación básica es igual a la de AndAR, aunque este SDK utiliza un motor de renderizado externo MIN3D, en lugar de cargar los objetos directamente. Min3D es una librería ligera 3D para Android que utiliza Java con OpenGL ES. Ofrece compatibilidad con la versión 1.5 de Android y OpenGL ES 1.0 y posteriores. Su documentación está integrada en bloques dentro del propio código.

### 2.3.4.3 Vuforia

Vuforia es un SDK desarrollado por Qualcomm, una empresa productora de chipsets para tecnología móvil. En 2010 la empresa lanzó algunas aplicaciones propias que hacían uso de tecnologías de RA, y finalmente ese mismo año anunció que ponía a disposición de los desarrolladores sus frameworks de desarrollo al que denominaron Vuforia. Está disponible para Android e iOS y se basa en el reconocimiento de imágenes basado en características especiales, por lo que también soporta marcadores naturales (targets) o RA sin marcadores. Además dispone de un plugin para interactuar con Unity3D y ofrece la posibilidad de crear botones virtuales para ampliar las vías de interacción con el usuario.

Una aplicación de RA basada en Vuforia estará integrada por los siguientes componentes fundamentales (*Figura.2-25*):

**Camera:**

Este módulo se asegura de que cada frame capturado pase al tracker. En este módulo se debe indicar cuándo la aplicación inicia la captura y cuando termina. El tamaño y formato de cada frame dependerá del dispositivo móvil utilizado.

**Image Converter:**

Este módulo convierte el formato de la cámara a un formato interoperable con OpenGL y para el tracking de los marcadores. Esta conversión incluye reducción de la tasa de muestreo con el fin de disponer de la imagen de la cámara en diferentes resoluciones.

**Tracker:**

Este módulo contiene los algoritmos de visión artificial que se encargan de la detección y rastreo de los objetos de cada frame. Diferentes algoritmos se encargan de la detección de nuevos “targets” o “markers” y de evaluar los botones virtuales. Los resultados son almacenados en un objeto de estado. Este módulo puede cargar múltiples conjuntos de objetos, pero nunca puede haber más de uno activo al mismo tiempo.

**Video Background Renderer:**

Este módulo procesa la imagen almacenada en el objeto de estado. El rendimiento de la representación de vídeo de fondo está optimizado para dispositivos específicos.

Todos estos componentes deben ser inicializados en nuestra aplicación. En cada frame se actualiza el objeto de estado y se llama a las funciones de renderizado. Así pues, nuestra aplicación siempre debe realizar estos tres paso:

1. Consultar el objeto de estado para comprobar nuevos targets o markers detectados.

2. Actualizar la lógica de la aplicación con los nuevos datos de entrada

3. Renderizar los elementos virtuales.

Los targets o marcadores son creados mediante un sistema online (Target Management System). Una vez creada la imagen que servirá como target o marcador, se accede a este sistema. Se crea un nuevo proyecto, y se sube la imagen. El sistema analiza la imagen y le asigna una calificación que indica la efectividad del marcador en función del número de características especiales detectadas por el sistema. El siguiente paso es convertir la imagen a formatos entendidos por la librería. El sistema nos devuelve dos archivos: un .xml

con la configuración del target o marcador y un archivo binario que contiene los datos rastreables.

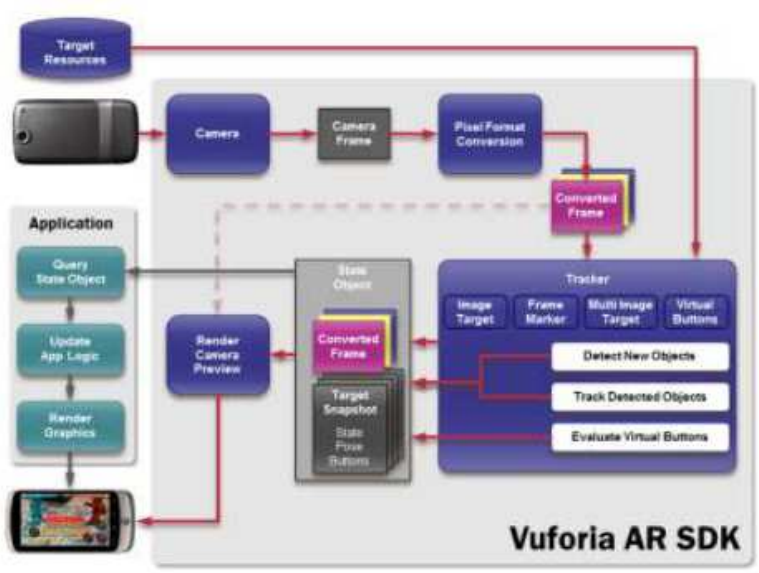


**Figura 2-25: Diagrama de flujo del SDK de Vuforia**

### 2.3.4.4 Metaio Mobile SDK

Metaio Mobile SDK es un SDK desarrollado por la empresa alemana Metaio. Hasta este año todas sus herramientas de desarrollo eran de pago, pero han liberado su SDK para móviles debido al éxito que estaba teniendo. La versión completa, de pago, incluye reconocimiento de caras y reconocedor de QR. Aunque la versión gratuita no incluye estas funcionalidades y que las aplicaciones desarrolladas deben incluir una marca de agua de la empresa, este SDK incluye un potente reconocedor de marcadores naturales, es decir, se pueden desarrollar aplicaciones “markerless”.

Esta empresa ganó la “tracking competition” del ISMAR 2011.

Otro aspecto muy llamativo a señalar de este framework, aunque no forma parte del estudio comparativo, es que incluye el concepto de gravedad en las tareas de reconocimiento así como de renderizado. Esto resulta un gran avance, pues facilita el reconocimiento en situaciones particulares como puede ser el reconocimiento de una ventana en concreto de un edificio de 100 plantas (Figura. 2-26).

**Figura 2-26: Ejemplo de aplicación de la gravedad en el reconocimiento**

Por otro lado el hecho de añadir gravedad a los objetos aumentados mejora enormemente la experiencia del usuario. Imaginemos una aplicación para que el usuario se pruebe pendientes, el hecho de que éstos se muevan como lo haría un objeto real con gravedad, aporta mucho realismo a la escena. Esto es sólo un simple ejemplo de lo que supone esta mejora. En la Figura. 2-27 se muestra otro ejemplo. Póngase atención a la orientación de la llama en función de la orientación del marcador:

**Figura 2-27: Ejemplo de la aplicación de la gravedad en los objetos virtuales.]**

Además de éste, la empresa dispone de otros dos SDK, de pago. El Metaio Web SDK, pensado para aplicaciones web, y enfocado principalmente al comercio electrónico. Y el Metaio PC SDK, pensado para aplicaciones de RA más robustas. Este SDK incluye el reconocimiento facial.

Existe una última aplicación desarrollada por esta empresa, que también ha sido utilizada en este trabajo. Es el Metaio creator, una herramienta pensada para generar espacios de RA rápidamente y sin necesidad de tener conocimientos de programación, todo funciona con una interfaz gráfica en la que se añaden marcadores y objetos 3D que la herramienta se encarga de transformar en una aplicación de RA. Este software es de pago, pero está disponible una versión Demo.

El metaio Mobile SDK está implementado de forma modular, al igual que los anteriores, de forma que divide la actividad de RA en tres componentes: Tracking, Captura y Renderizado:

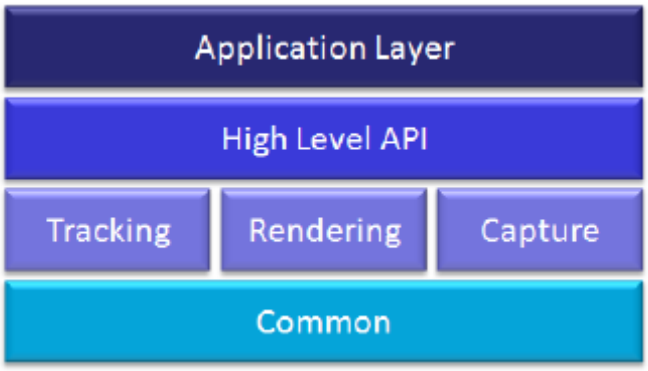


**Figura 2-28: Estructura del Metaio Mobile SDK**

En la versión gratuita la etapa de tracking está oculta bajo una aplicación estándar configurable mediante un archivo de configuración .XML. Este archivo, que puede obtenerse a partir de nuestro target con la versión Demo de Metaio Creator de una manera parecida a lo que ocurría con Vuforia, contiene los parámetros de configuración del target, tamaño y nombre de la imagen .png utilizada entre otras.

### 2.3.4.5 ArUco

Esta librería comenzó su desarrollo en C++ y posteriormente fue portada para la plataforma Android.

Las principales características son:

- Detección de marcadores.
- Detección de tableros de marcadores.
- Creación de marcadores.
- Estimación pose de objetos 3D.
- Interacción con los objetos 3D.
- Animación de objectos 3D.

#### 2.3.4.5.1 Proceso de Detección en Aruco

El proceso de detección de un marcador en la librería ArUco es el siguiente:

- Aplicar un threshold adaptativo para obtener los bordes en una imagen.
- Encontrar los contornos. Después de eso, no solamente el marcador real puede ser detectado, sino que otros bordes no deseados también pueden serlo. Para filtrarlos en el sistema:
    - Eliminar los bordes con un número pequeño de puntos.

- Aproximar el contorno a un polígono y verificar que tiene exactamente 4 esquinas.

- Establecer las esquinas en dirección contraria a las agujas del reloj.
- Eliminar los rectángulos muy cercanos. Esto es debido a que el threshold adaptativo detecta la parte interna y externa del borde de un marcador. En este punto, el borde más externo se ha almacenado.
- Identificación de un marcador
  - Eliminar la perspectiva para obtener una vista frontal del marcador.
  - Aplicar al área un threshold mediante el algoritmo de Otsu's, que asume una distribución bimodal en el marcador.
  - Identificar el marcador interno. Si es un marcador, tiene un código interno. El marcador se divide en una matriz 6x6, de las cuales las celdas internas 5x5 contienen la información del id. El resto corresponden al borde externo de color negro.
- Para un marcador válido, las esquinas se refinan mediante interpolación.

Este proceso se muestra en la siguiente imagen.

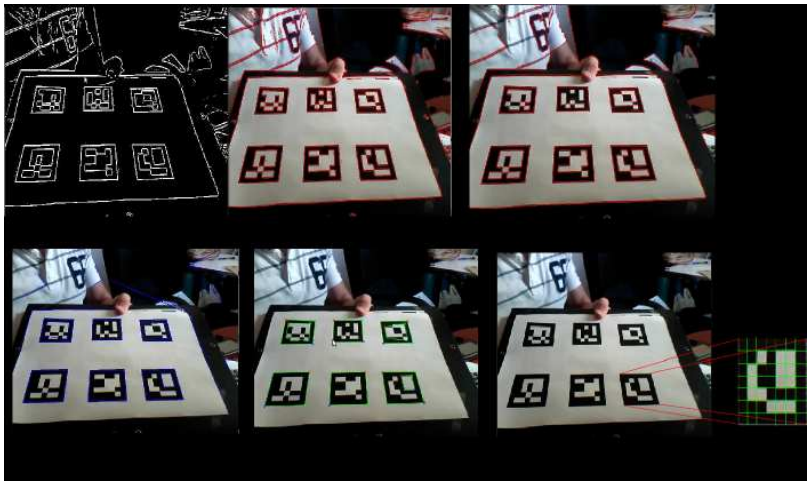

**Figura 2-29: Proceso de detección en la librería ArUco**

#### 2.3.4.5.2 Codificación de un Marcador

Cada marcador tiene un código interno de 5 palabras de 5 bits cada una. Esta codificación empleada es una ligera modificación del código *Hamming*. En total, cada palabra solo tiene 2 bits de información de los 5 bits empleados. Los otros 3 son empleados para detección de errores. Como consecuencia, puede haber hasta 1024 identificadores diferentes.

La mayor diferencia con el código Hamming es que el primer bit (bits de paridad 3 y 5) se han invertido. De esta manera, el id 0 (que sería 00000 con el código Hamming) se convierte en 10000 en la codificación de la librería. La idea es la prevención de utilizar un

rectángulo completamente negro como marcador válido para poder reducir falsos positivos con los objetos del entorno.

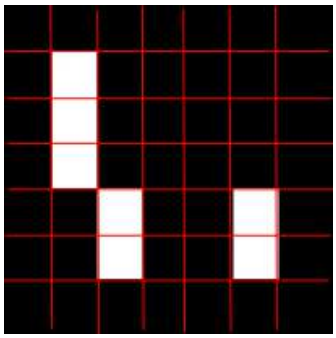

**Figura 2-30: Codificación de un marcador en ArUco**

### 2.3.5 Elección del SDK de RA

Después de analizar y probar las cinco librerías de realidad aumentada descritas en el apartado anterior se llegó a la conclusión de que la mejor opción para el proyecto era la utilización de la librería OpenCv de manera directa, siguiendo una gestión similar al de la librería ArUco (y utilizando sus clases de detección de marcadores). Esta decisión se basa en los siguientes puntos:

- Distancia del marcador: mayor distancia de reconocimiento, entre 3’5 y 4 metros.
- Tamaño del marcador: menor tamaño de marcador, 7’5x7’5 centímetros.
- Identificación del marcador: el marcador debe codificar un entero, para poder diferenciar los distintos tipos de dispositivos y acciones domóticas. ArUco permite codificar enteros con un rango entre 1 y 1024.
- Creación de marcadores: ArUco permite crear hasta 1024 marcadores diferentes de manera sencilla a través de su proyecto en Android.
- Flexibilidad del sistema y utilización de objetos 3d: la mayoría de los SDKs están orientados a modelado de objetos 3D que incrementan la complejidad de desarrollo frente a modelos clásicos 2D. OpenCV es más flexible ya que permite gestionar modelos 2D de forma sencilla.

## 2.4 VISIÓN ARTIFICIAL

La visión artificial se define como el campo de la inteligencia artificial que, mediante la utilización de técnicas adecuadas, permite adquirir, procesar y analizar cualquier tipo de información obtenida a través de imágenes digitales. Es decir, la visión artificial es la aplicación de la visión por computador a la industria con el objetivo de controlar un sistema automático usando información visual.

El término "procesamiento de imagen" se refiere a aplicar un conjunto de técnicas con el objetivo de mejorar su calidad o facilitar la búsqueda de información, en otras palabras, estamos transformando la imagen inicial en otra u otras similares a la inicial pero con algunas características modificadas. Un ejemplo sería la manipulación del contraste o el brillo, la aplicación de filtros para la eliminación de ruido, etc.

Volviendo al tema de la visión por computador, y una vez comprendidos estos conceptos, se pueden exponer los pasos de este tipo de proyectos.


**Figura 2-31: Etapas de un sistema de Visión por Computador**

En la figura anterior se pueden ver las fases que componen un proyecto de visión por computador junto a un ejemplo ilustrativo de un sistema que, mediante la lectura de la

matrícula y posterior búsqueda en una base de datos, dispone a elección del operario el dejar entrar o salir al vehículo accionando la barrera que le impide el paso. Para ello le informa si esa matrícula está o no en la base de datos.

El proceso parte del objeto a detectar, el cual es captado por un sistema de cámaras de muy diversos tipos. En el interior de estas cámaras es donde se forma la imagen que, posteriormente, será preprocesada para mejorar su calidad en caso de ser necesario.

Una vez adquirida la imagen se procede a su análisis mediante la extracción de las características más relevantes o, simplemente, aquellas que necesitemos para reconocer y localizar el objeto captado. Con la información obtenida se interpreta la imagen y, finalmente, se lleva a cabo la toma de decisiones.

Cabe destacar que la extracción de características es una de las fases críticas en el proceso, pues de ella depende la correcta o incorrecta actuación final. Este paso ha sido estudiado por gran cantidad de investigadores, llegándose a desarrollar diversos métodos pero, habitualmente, todos ellos constan de una detección de contornos seguida de una división de la imagen en varias regiones u objetos, proceso conocido como segmentación.

### 2.4.1 OpenCV

OpenCV (*Open Source Computer Vision*) es una librería que contiene funciones de Visión por Computador en tiempo real. La primera versión alfa fue originalmente desarrollada por Intel en enero de 1999 [21].

OpenCV es multiplataforma, es decir, las interfaces C, C++, Python y Java que contiene, funcionan a la perfección en diversos Sistemas Operativos: Windows, GNU/Linux, Android y Mac OS X.

Estas librerías contienen más de 500 funciones optimizadas, que abarcan distintas áreas en el proceso de Visión por Computador, como el reconocimiento de objetos, calibración de cámaras, visión estéreo y visión robótica avanzada. Gracias a esta diversidad, y a que OpenCV fue publicada bajo la licencia BSD, que permite su uso libre tanto para propósitos comerciales como de investigación, se ha utilizado en infinidad de aplicaciones, desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere el reconocimiento de objetos.

Las librerías OpenCV se dividen en cinco grandes grupos:

- *CXCORE*: aquí se encuentran las estructuras y algoritmos básicos que usan las demás funciones, como la suma, media, operaciones binarias, etc.
- *CV*: en este grupo están implementadas las funciones principales de procesamiento de imágenes.
- *HighGUI*: en él se encuentra todo lo relacionado con la interfaz gráfica de OpenCV, brinda herramientas para trabajar con imágenes y vídeos guardados en archivos u obtenidos directamente desde cámaras.
- *ML*: aquí se encuentran algoritmos de aprendizaje y clasificadores.
- *CvAux*: que contiene algoritmos experimentales

## 2.5 SISTEMAS DE DOMÓTICA

### 2.5.1 Introducción

Se entiende como domótica el conjunto de sistemas capaces de automatizar una vivienda. De este modo, se le aporta servicios de gestión energética, seguridad, bienestar y comunicación que pueden ser integrados por medio de redes interiores y exteriores de comunicación. Estas comunicaciones pueden ser inalámbricas o cableadas, cuyo control goza de cierta ubicuidad desde dentro y fuera del hogar. De forma sencilla, se podría definir como:

“La integración de la tecnología en el diseño inteligente de un recinto cerrado”

El término, viene de la unión de dos palabras como ‘domus’ (que quiere decir, casa, en latín) y tica (de automática, palabra que en latín quiere decir ‘que funciona por sí sola’). En España la domótica tiene presencia mediante multitud de empresas, algunas con más de 12 años en el mercado.

**Figura 2-32: Ejemplo de Casa Domótica**

### 2.5.2 Servicios en Sistemas de Domótica

Principalmente, los servicios que ofrecen este tipo de sistemas se pueden clasificar en cinco ámbitos.

#### 2.5.2.1 Ahorro Energético

El ahorro energético no es algo tangible, sino un concepto al que se puede llegar de muchas maneras. En muchos casos no es necesario sustituir los aparatos o sistemas del hogar por otros que consuman menos sino una gestión eficiente de los mismos mediante distintos mecanismos:

- Climatización: programación y zonificación.
- Gestión eléctrica:
  - Racionalización de cargas eléctricas: desconexión de equipos de uso no prioritario en función de cierto consumo eléctrico en cierto tiempo.
  - Gestión de tarifas: derivando el funcionamiento de los aparatos de mayor consumo energético a ciertas horas del día en el cual la tarifa es más barata.
- Uso de energías renovables.

#### 2.5.2.2 Confort

El confort, conlleva todas las actuaciones que se puedan llevar a cabo que mejoren las condiciones de comodidad de una vivienda. Dichas actuaciones, pueden ser de carácter activo, pasivo o mixto:

- Iluminación:
  - Apagado general de todas las luces de la vivienda.
  - Automatización del apagado o encendido en cada foco de luz.
  - Regulación de la iluminación según necesidades ambiente.
- Automatización: Los distintos sistemas/instalaciones/equipos dotándolos de un control eficiente y de fácil manejo.
- Integración del portero al teléfono o del video portero al televisor.
- Control vía internet.
- Gestión Multimedia del ocio electrónico.
- Generación de programas de forma sencilla para el usuario.

#### 2.5.2.3 Seguridad

Consiste en una red de seguridad encargada de proteger tanto los bienes patrimoniales como la seguridad personal.

- Alarmas de intrusión (anti-intrusión): se utilizan para detectar o prevenir la presencia de personas extrañas en una vivienda o edificio.
  - Detección de un posible intruso (Detectores volumétricos o perimetrales)

  - Cierre de persianas puntual y seguro
  - Simulación de presencia
- Alarmas de detección de incendios, fugas de gas, escapes de agua, concentración de monóxido en garajes cuando se usan vehículos de combustión.
- Alerta médica, tele asistencia.
- Acceso a cámaras IP.

#### 2.5.2.4 Comunicaciones

Los sistemas o infraestructuras de comunicaciones que posee el hogar.

- Ubicuidad en el control tanto externo como interno, control remoto desde Internet, PC, mandos inalámbricos (p.ej. PDA con WiFi), aparellaje eléctrico.
- Tele asistencia
- Tele mantenimiento
- Informes de consumo y costes
- Transmisión de alarmas.
- Intercomunicaciones.

#### 2.5.2.5 Accesibilidad

Bajo este epígrafe se incluyen las aplicaciones o instalaciones de control remoto del entorno que favorecen la autonomía personal de personas con limitaciones funcionales, o discapacidad.

El concepto "diseño" para todos es un movimiento que pretende crear la sensibilidad necesaria para que al diseñar un producto o servicio se tengan en cuenta las necesidades de todos los posibles usuarios, incluyendo las personas con diferentes capacidades o discapacidades, es decir, favorecer un diseño accesible para la diversidad humana.

La inclusión social y la igualdad son términos o conceptos más generalistas y filosóficos. La domótica aplicada a favorecer la accesibilidad es un reto ético y creativo pero sobre todo es la aplicación de la tecnología en el campo más necesario, para suplir limitaciones funcionales de las personas.

El objetivo no es que las personas con discapacidad puedan acceder a estas tecnologías, porque las tecnologías en si no son un objetivo, sino un medio. El objetivo de estas tecnologías es favorecer la autonomía personal. Los destinatarios de estas tecnologías son todas las personas, ya que por enfermedad o envejecimiento, todos somos o seremos discapacitados, más pronto o más tarde.

## 2.5.3 El Sistema Domótico

### 2.5.3.1 Dispositivos

Los dispositivos que componen un sistema de domótica tradicional, incluyen principalmente tres tipos:

- Controladores.
- Sensores.
- Actuadores.

### 2.5.3.2 Arquitectura

Desde el punto de vista donde reside la inteligencia del sistema de domótica, hay varias arquitecturas distintas.

- Arquitectura Centralizada: un controlador centralizado recibe información de múltiples sensores y, una vez procesada, genera las órdenes oportunas para los actuadores.
- Arquitectura Distribuida: toda la inteligencia del sistema está distribuida por todos los módulos sean sensores o actuadores. Suele ser típico de los sistemas de cableado en bus, o redes inalámbricas.
- Arquitectura mixta: sistemas con arquitectura descentralizada en cuanto a que disponen de varios pequeños dispositivos capaces de adquirir y procesar la información de múltiples sensores y transmitirlos al resto de dispositivos distribuidos por la vivienda, p.ej. aquellos sistemas basados en Zigbee y totalmente inalámbricos.

### 2.5.3.3 Elementos de una Instalación Domótica

- Central de gestión
- Sensores
- Actuadores
- Soportes de comunicación
- Aparatos terminales

### 2.5.4 Asociaciones Internacionales y Españolas de Domótica

**CENELEC:**

La Comisión CENELEC/ENTR/e-Europe/2001-03 es la encargada de elaborar normas a nivel internacional y la organización que ha promocionado el Smart House Forum.

**CEDOM:**

Asociación Española de Domótica. Su objetivo principal es la promoción de la Domótica. Se trata del foro nacional en el que se reúnen todos los agentes del sector en España: fabricantes de productos domóticos, fabricantes de sistemas, instaladores, integradores, arquitecturas e ingenierías, centros de formación, universidades, centros tecnológicos.

**LonUsers España**

Asociación de usuarios de la tecnología LonWorks, siendo creada por la iniciativa de empresas líderes en los diferentes sectores de aplicación de la tecnología LonWorks (domótica, inmótica, control industrial y de transporte).

**AENOR**

Asociación que ha creado también una Subcomisión del Hogar Digital, dentro de la Comisión 133 (AEN/CTN 133 Telecomunicaciones) a fin de definir estándares.

### 2.5.5 Hogar Digital

El hogar digital es una vivienda que a través de equipos y sistemas, y la integración tecnológica entre ellos, gracias a la domótica, ofrece a sus habitantes funciones y servicios que facilitan la gestión y el mantenimiento del hogar, aumentan la seguridad; incrementan el confort; mejoran las telecomunicaciones; ahorran energía, costes y tiempo, y ofrecen nuevas formas de entretenimiento, ocio y otros servicios dentro de la misma y su entorno sin afectar a las casas normales.

#### 2.5.5.1 Diferencias entre Casa Domótica y Hogar Digital

Una vivienda domótica dispone de un gran número de equipos y sistemas, principalmente autónomos, a los que hay que sumar diferentes redes, como la telefonía, las redes de datos (cableadas e inalámbricas), la televisión, electrodomésticos, equipamiento de audio y video, calefacción, aire-condicionado, seguridad, riego, iluminación, etc.

Para convertirse en un hogar digital, a una casa domótica le faltaría la convergencia de las comunicaciones, la informática y el entretenimiento gracias a las redes de banda ancha es una tendencia consolidada a nivel mundial.

Es necesario resaltar que el Hogar Digital no consiste simplemente en la instalación de dispositivos para controlar determinadas funciones de las viviendas, tales como alarmas, iluminación, climatización, etc.

### 2.5.6 Descripción del Sistema Domótico.

En este apartado se van a describir todos los elementos del sistema domótico con el que interactuará la aplicación móvil desarrollada en el proyecto. Para ello, se iniciará con una breve introducción en la que se dará pie al sistema domótico, para seguidamente, describir los requisitos extraídos del sistema domótico.

Finalmente, se terminará con una breve descripción técnica de cada uno de los elementos del sistema domótico.

#### 2.5.6.1 Introducción.

Dado que el sistema domótico se establecerá en el propio hogar de la persona de movilidad reducida, el sistema domótico deberá ser fiable, robusto y sencillo de utilizar.

A pesar de todo ello, como cualquier sistema domótico necesitará de un mantenimiento y de una cierta supervisión por parte de un administrador. Es por ello, que uno de los requisitos que se establecerán es el de la habilitación de un método de acceso remoto al servidor.

Se ha utilizado el estándar KNX para el desarrollo de este sistema. KNX es el estándar mundial para el control de casas y edificios. Aprobado como Estándar Internacional (ISO/IEC 14543-3), así como en el Estándar Europeo (CENELEC EN 50090 y CEN EN 13321-1) y Estándar en China (GB/Z 20965). Se garantiza que los productos KNX hechos

por diferentes fabricantes pueden ser combinados. La propia marca registrada KNX garantiza la interoperabilidad y el "*interworking*".

Cabe destacar, que este estándar está basado en otros con más de 15 años de experiencia en el mercado. Entre otros, los sistemas predecesores de KNX son: EIB, EHS y BatiBUS.

Los medios de transmisión para este tipo de infraestructuras son de diversos tipos, van desde el par trenzado al IP/Ethernet. Sobre el medio de transmisión, se han de encontrar todos los dispositivos interconectados.

Los dispositivos conectados al bus, tanto sensores (en caso de necesitarlos), como actuadores, son utilizados para el control de equipamiento de gestión de edificios en todas las aplicaciones posibles: iluminación, persianas, sistemas de seguridad, gestión energética, calefacción, sistemas de ventilación y aire acondicionado, sistemas de supervisión y señalización, interfaces a servicios y sistemas de control de edificios etc.

Todas estas funciones pueden ser controladas y señalizadas utilizando un sistema uniforme sin la necesidad de centros de control adicionales.


**Figura 2-33: Resumen de las Funciones del Sistema Domótico**

### 2.5.6.2 Requisitos del Sistema Domótico.

Dado que el sistema ha de realizar una serie de acciones que le vienen en forma de petición 'HTTP *request'*, el servidor, deberá ser capaz de realizar acciones de procesado, enrutado y respuesta. Con la tecnología actual en sistemas domóticos, el proceso de enrutado será desempeñado por un bus autónomo energéticamente. De éste modo, cuando le llegue un '*request*' al servidor, éste, lo procesará, identificará el dispositivo al que va dirigida dicha acción y finalmente mandará el mandato al dispositivo de la red a través del BUS de direccionamiento. Éste, es un punto importante, dado que el servidor ha de estar integrado con el BUS para que sea capaz de recibir y enviar la orden de la aplicación móvil.

No obstante, no termina ahí el trabajo del servidor. Éste, ha de procesar una respuesta del dispositivo (actuador en nuestro caso), diciéndole si el mandato enviado ha podido ser ejecutado exitosamente o no. Por último, dado que el servidor conoce si el mandato enviado al dispositivo ha tenido un resultado exitoso, se lo ha de comunicar a la aplicación móvil. Finalmente, la aplicación podrá informar del resultado de la petición realizada por el usuario.

Por lo tanto, se puede entender que el servidor ha de ser capaz de recibir peticiones 'HTTP *request'*, enviar peticiones y además procesar los posibles mensajes de respuesta que le lleguen desde los dispositivos a través del BUS y comunicárselo a la aplicación móvil.

Dado que ya se ha realizado una explicación de los requisitos del servidor domótico y BUS de direccionamiento (a mayor carga del mismo), queda realizar una descripción de los requisitos que los dispositivos tienen.

Como el servidor domótico lanza una serie de mandatos sobre los mismos, éstos deben de estar preparados para las acciones posibles del elemento que controlan.

Por ejemplo, el dispositivo encargado de controlar la luz, tendrá que conocer sus acciones (encender o apagar la luz) y su estado (encendido o apagado). De este modo, cuando el servidor domótico le lance la petición de encender o apagar, éste estará listo y sabrá contestar éxito o no en función de que ésta haya sido realizada.

Para facilitar la comprensión del sistema que se desea obtener, que se expone un diagrama de alto nivel.

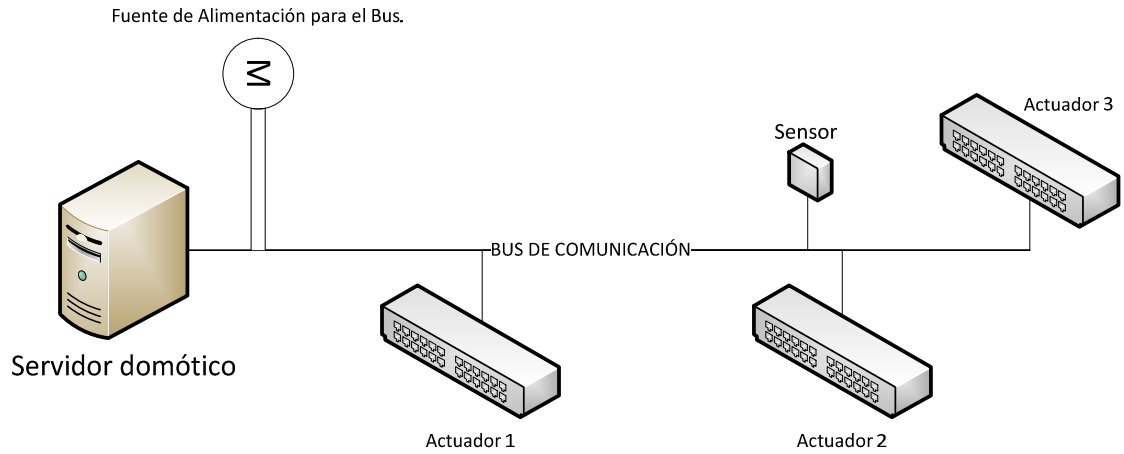


**Figura 2-34: Diagrama del Sistema Domótico Asistencial**

### 2.5.6.3 Infraestructura del sistema.

El sistema domótico necesario para este proyecto, cuenta con los elementos básicos de un sistema de domótica tradicional.

Un servidor domótico (cliente IP) que realizará el trabajo de enviar los mandatos a los dispositivos de su red. Además, será necesario un BUS de datos que sea capaz de interconectar el servidor domótico (ó central IP) con los dispositivos que se encuentren domotizados.

Es decir, deberá ser capaz de enviar de un punto a otro las peticiones de los dispositivos. Además, deberá contar con los dispositivos que automatizan mediante un 'actuador' las acciones a realizar.

Por ejemplo, el servidor domótico conectado a una red, recibirá de la aplicación móvil un mandato por medio de un HTTP *request*. A partir de ese momento, el servidor, enviará dicha acción al dispositivo que corresponda, la acción a acometer por medio de un BUS de comunicación. Por último el dispositivo, deberá automatizar la acción recibida.

A continuación, se enumeran los dispositivos necesarios:

- Cliente IP.
- BUS de comunicaciones.
- Fuente de alimentación para el BUS de comunicaciones.
- Actuador y Sensor.
- Comunicador USB.

#### 2.5.6.3.1 Servidor domótico

El primer elemento que se ha querido especificar es el servidor domótico. Se trata de una central IP capaz de atender peticiones HTTP *request*. El modelo que se ha seleccionado he implantado es el que se expone a continuación:

- Dispositivo: central IP
- Objetivo: atender peticiones 'HTTP'.
- Código del producto: IPZ 1000 REG

**Figura 2-35: Sistema Domótico - Central IP**

#### 2.5.6.3.2 Fuente de alimentación (Bus)

El siguiente elemento que se ha querido especificar es la fuente de alimentación que dará corriente al BUS que unirá la central IP con el dispositivo de comunicación. El modelo elegido para ello, ha sido:

- Dispositivo: fuente de Alimentación (BUS).
- Objetivo: enrutar el tráfico entre la central IP y los módulos de comunicación de los dispositivos.

- Código del producto: 2002 REG
- Características: cuenta con protección de sobrecargas y cortocircuitos del bus.

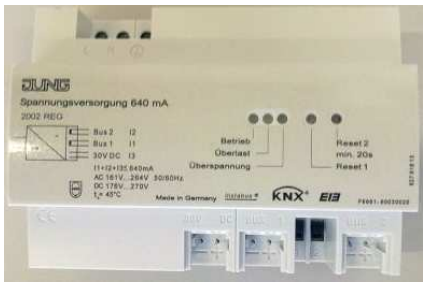


**Figura 2-36: Sistema Domótico - Fuente de Alimentación**

### 2.5.6.3.3 Actuador

El siguiente elemento que se ha querido especificar es el actuador, será el dispositivo encargado de mandar las acciones a realizar a los dispositivos. El modelo elegido para ello, ha sido:

- Dispositivo: actuador de 8 salidas.
- Objetivo: mandar las acciones a realizar al dispositivo final (detector de movimiento, detector de luz… etc).
- Código del producto: 2308.16 REGGM
- Características: se trata de un actuador de 16 A (amperios).

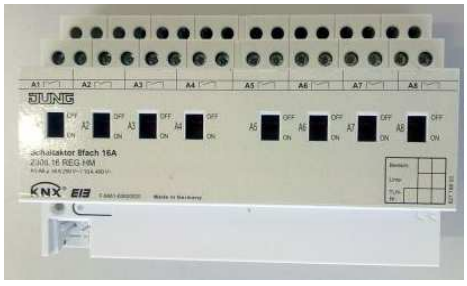


**Figura 2-37: Sistema Domótico - Actuador**

### 2.5.6.3.4 Sensor.

El siguiente elemento que se ha querido especificar es el sensor, será el dispositivo encargado de conocer el estado del dispositivo. El modelo elegido para ello, ha sido:

- Dispositivo: sensor de luz.
- Objetivo: conocer el estado de la luz.
- Código del producto: LS 3180-1

- Características: requiere de un acoplador de bus cod.:2070U.

**Figura 2-38: Sistema Domótico - Sensor**

#### 2.5.6.3.5 Módulo de comunicación USB.

El siguiente elemento que se ha querido especificar es el módulo de comunicación USB, será el dispositivo encargado de permitir la conexión de un detector o dispositivo y el ordenador o servidor domótico por medio de un puerto USB. El modelo elegido para ello, ha sido:

- Dispositivo: módulo de comunicación USB.
- Objetivo: manda las acciones a realizar al dispositivo final (Detector de movimiento, Detector de luz… etc).
- Código del producto: 2130 USB
- Características: contiene un puerto USB 2.0.


**Figura 2-39: Sistema Domótico - Módulo de Comunicación USB**

#### 2.5.6.3.6 Otros Componentes Necesarios

Finalmente, se exponen una serie de materiales algo triviales o no, para el montaje de la instalación. Como se puede observar en la ilustración, fueron necesarios terminales de conexión, una serie de marcos para el montaje tanto vertical como horizontal y finalmente el cable del BUS:

- Dispositivo: materiales varios.
- Objetivo: completar la instalación.
- Código del producto: Terminales de Conexión (2050RT SW), marco para montaje (FD981WW), cable BUS J-Y
- Características: contiene un acoplador para el bus de código 2070U.

**Figura 2-40: Sistema Domótico - Otros Componentes**

# 3. ANÁLISIS

Este capítulo está centrado en la especificación de la aplicación, definiendo los requisitos y las funcionalidades que debe ofrecer. Se detallan los requisitos de usuario y de software, los casos de uso de la aplicación y por ultimo los diagramas de actividad del sistema.

## 3.1 REQUISITOS DE USUARIO

En este capítulo se detallan los requisitos de usuario a través un formato tipo tabla:

**Tabla 3-1: Requisito de Usuario – REQ_USER_01**

| **REQ_USER_01** | Aplicación móvil | | |
|---|---|---|---|
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La aplicación deberá ejecutarse en un teléfono móvil | | |

**Tabla 3-2: Requisito de Usuario – REQ_USER_02**

| **REQ_USER_02** | Reconocer marcadores | | |
|---|---|---|---|
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La aplicación reconocerá imágenes mediante la cámara | | |

**Tabla 3-3: Requisito de Usuario – REQ_USER_03**

| **REQ_USER_03** | Imagén identificará dipositivos | | |
|---|---|---|---|
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La imagen reconocida permitirá diferenciar entre distintos dispositivos del sistema domótico | | |

**Tabla 3-4: Requisito de Usuario – REQ_USER_04**

| **REQ_USER_04** | Ejecutar una acción sobre el dipositivo | | |
|---|---|---|---|
| **Prioridad** | Alta | **Necesidad** | Esencial |

| | | | |
|---|---|---|---|
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Una vez reconocido el dispositivo se podrá realizar una acción sobre él | | |

**Tabla 3-5: Requisito de Usuario – REQ_USER_05**

| | | | |
|---|---|---|---|
| **REQ_USER_05** | Fijar la cámara un tiempo mínimo | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Hay que fijar el móvil unos segundos ante la imagen para ejecutar la accion | | |

**Tabla 3-6: Requisito de Usuario – REQ_USER_06**

| | | | |
|---|---|---|---|
| **REQ_USER_06** | La imagen puede ser candidata o estar seleccionada | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Una vez reconocida la imagen, puede ser candidata (solo reconocida) o estar seleccionada (si ya ha comenzado el proceso de ejecutar una acción) | | |

**Tabla 3-7: Requisito de Usuario – REQ_USER_07**

| | | | |
|---|---|---|---|
| **REQ_USER_07** | Emitir mensajes de información por voz | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Informar al usuario mediante mensajes de voz | | |

**Tabla 3-8: Requisito de Usuario – REQ_USER_08**

| | | | |
|---|---|---|---|
| **REQ_USER_08** | Mostrar mensajes de información por pantalla | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Informar con un mensaje en la pantalla | | |

**Tabla 3-9: Requisito de Usuario – REQ_USER_09**

| **REQ_USER_09** | Elegir una acción por reconocimiento de voz | | |
|---|---|---|---|
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Una acción se puede ejecutar reconociendo la voz | | |

**Tabla 3-10: Requisito de Usuario – REQ_USER_10**

| **REQ_USER_10** | Elegir una acción pulsando la pantalla | | |
|---|---|---|---|
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Una acción se puede ejecutar mediante pulsando en la pantalla | | |

**Tabla 3-11: Requisito de Usuario – REQ_USER_11**

| **REQ_USER_11** | Configuración del tipo de interfaz | | |
|---|---|---|---|
| **Prioridad** | Media | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | El usuario podrá seleccionar si desea mensajes por pantalla y/o voz y seleccionar acciones por pantalla y/o voz | | |

**Tabla 3-12: Requisito de Usuario – REQ_USER_12**

| **REQ_USER_12** | Configuración del servidor domótico | | |
|---|---|---|---|
| **Prioridad** | Baja | **Necesidad** | Deseable |
| **Estabilidad** | Inestable | **Verificabilidad** | Alta |
| **Descripción** | La dirección IP y el puerto del servidor domótico podrán configurarse por el usuario | | |

**Tabla 3-13: Requisito de Usuario – REQ_USER_13**

| **REQ_USER_13** | Tiempo mínimo de reconocimiento configurable | | |
|---|---|---|---|
| **Prioridad** | Baja | **Necesidad** | Deseable |
| **Estabilidad** | Inestable | **Verificabilidad** | Alta |

| | |
|---|---|
| **Descripción** | El tiempo mínimo necesario para reconocer un marcador podrá configurarse por el usuario |

## 3.2 DIAGRAMAS DE CASOS DE USO

La interacción del usuario con el sistema se define en los siguientes casos de uso:

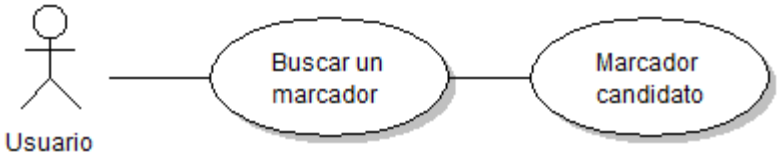


**Figura 3-1: Caso de Uso – Buscar un Marcador**

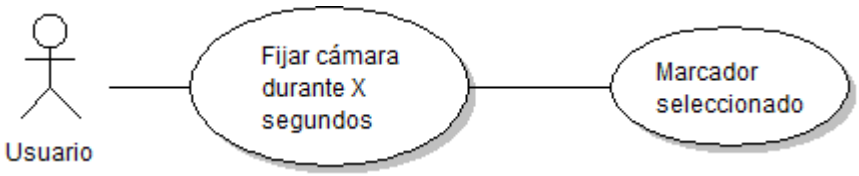


**Figura 3-2: Caso de Uso – Fijar la Cámara el Tiempo Mínimo**

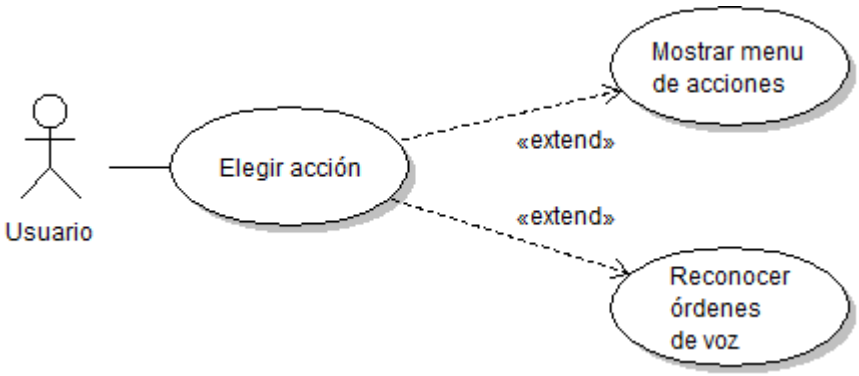


**Figura 3-3: Caso de Uso – Elegir una Acción**

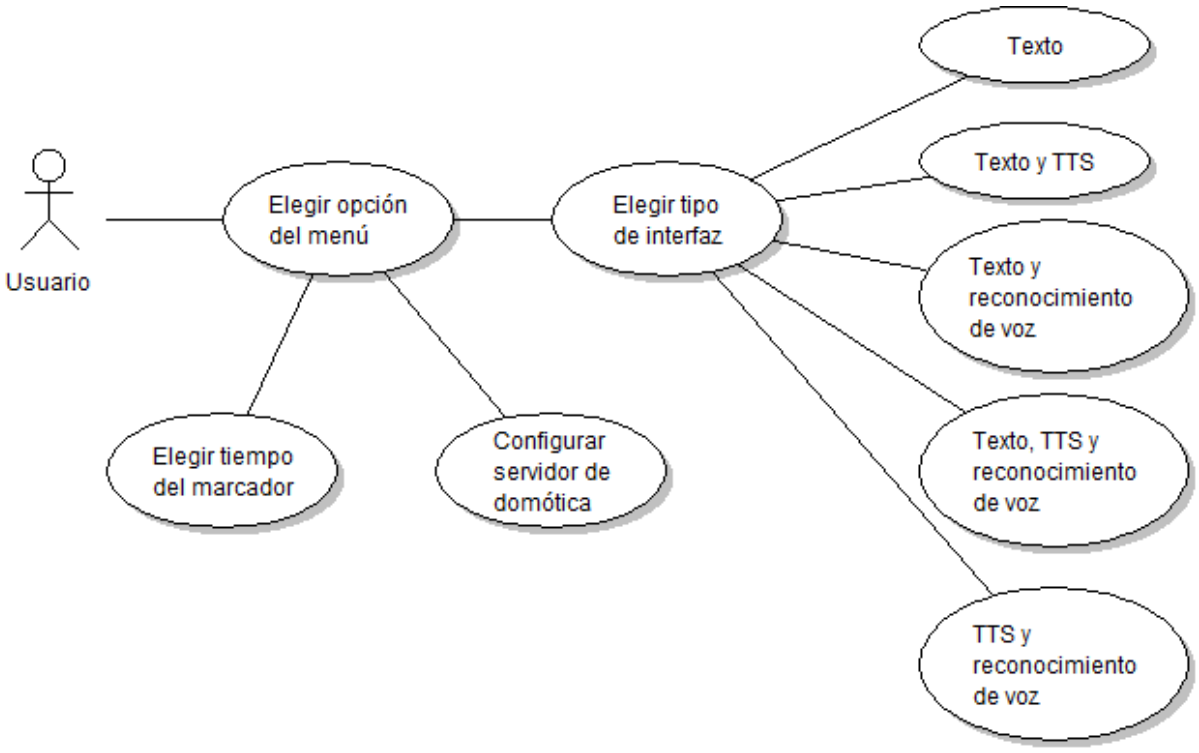


**Figura 3-4: Caso de Uso – Elegir una Opción del Menú**

## 3.3 REQUISITOS DEL SOFTWARE

A continuación se detallan los requisitos de software con el mismo formato que los requisitos de usuario.

**Tabla 3-14: Requisito Software – REQ_SW_01**

| | | | |
|---|---|---|---|
| **REQ_SW_01** | Aplicación android | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La aplicación deberá ejecutarse en un teléfono móvil con sistema operativo Android | | |

**Tabla 3-15: Requisito Software – REQ_SW_02**

| | | | |
|---|---|---|---|
| **REQ_SW_02** | Uso de la cámara | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Se mostrará al usuario en todo momento el video capturado por la cámara | | |

**Tabla 3-16: Requisito Software – REQ_SW_03**

| | | | |
|---|---|---|---|
| **REQ_SW_03** | Reconocer marcadores | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La aplicación reconocerá marcadores | | |

**Tabla 3-17: Requisito Software – REQ_SW_04**

| | | | |
|---|---|---|---|
| **REQ_SW_04** | Tamaño marcador | | |
| **Prioridad** | Baja | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | El tamaño del marcador será de aproximadamente 7,5x7,5 cm | | |

**Tabla 3-18: Requisito Software – REQ_SW_05**

| | |
|---|---|
| **REQ_SW_05** | Distancia marcador |

| | | | |
|---|---|---|---|
| **Prioridad** | Baja | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La aplicación capturará marcadores hasta una distancia de 3,5-4 metros. | | |

**Tabla 3-19: Requisito Software – REQ_SW_06**

| | | | |
|---|---|---|---|
| **REQ_SW_06** | Identificador marcador | | |
| **Prioridad** | Media | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Cada marcador se codificará como un entero con un rango de [1, 1024] | | |

**Tabla 3-20: Requisito Software – REQ_SW_07**

| | | | |
|---|---|---|---|
| **REQ_SW_07** | Asociación marcador – dispositivo | | |
| **Prioridad** | Media | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | El entero asociado al marcador identificará unívocamente al dispositivo del servidor de domótica | | |

**Tabla 3-21: Requisito Software – REQ_SW_08**

| | | | |
|---|---|---|---|
| **REQ_SW_08** | Tiempo mínimo reconocimiento | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | El usuario deberá fijar la cámara un número de segundos determinado antes de poder procesar el marcador | | |

**Tabla 3-22: Requisito Software – REQ_SW_09**

| | | | |
|---|---|---|---|
| **REQ_SW_09** | Marcadores múltiples | | |
| **Prioridad** | Baja | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | En caso de reconocer varios marcadores, el marcador elegido será el más cercano al centro | | |

**Tabla 3-23: Requisito Software – REQ_SW_10**

| | | | |
|---|---|---|---|
| **REQ_SW_10** | Procesamiento del marcador | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La aplicación procesará el marcador a partir del requisito REQ_SW_08 | | |

**Tabla 3-24: Requisito Software – REQ_SW_11**

| | | | |
|---|---|---|---|
| **REQ_SW_11** | Listado de acciones | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La aplicación mostrará al usuario el listado de acciones asociado al dispositivo reconocido | | |

**Tabla 3-25: Requisito Software – REQ_SW_12**

| | | | |
|---|---|---|---|
| **REQ_SW_12** | Listado de acciones por menu de opciones | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Las acciones se mostrarán al usuario en el menú de opciones de la aplicación | | |

**Tabla 3-26: Requisito Software – REQ_SW_13**

| | | | |
|---|---|---|---|
| **REQ_SW_13** | Listado de acciones por voz | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La aplicación dictará las acciones al usuario mediante *Text To Speech* | | |

**Tabla 3-27: Requisito Software – REQ_SW_14**

| | | | |
|---|---|---|---|
| **REQ_SW_14** | Selección de la acción | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | El usuario podrá seleccionar una acción entre el listado de acciones mostrado | | |

**Tabla 3-28: Requisito Software – REQ_SW_15**

| | | | |
|---|---|---|---|
| **REQ_SW_15** | Selección de la acción mediante reconocimiento de voz | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | El usuario podrá elegir la acción a ejecutar a través del menú de opciones | | |

**Tabla 3-29: Requisito Software – REQ_SW_16**

| | | | |
|---|---|---|---|
| **REQ_SW_16** | Selección de la acción por menú de opciones | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | El usuario podrá elegir la acción a ejecutar mediante reconocimiento de voz. | | |

**Tabla 3-30: Requisito Software – REQ_SW_17**

| | | | |
|---|---|---|---|
| **REQ_SW_17** | Conexión al servidor de domótica | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Una vez elegida la acción, se ejecutará una petición HTTP hacia el servidor de domótica. | | |

**Tabla 3-31: Requisito Software – REQ_SW_18**

| | | | |
|---|---|---|---|
| **REQ_SW_18** | Conexión al servidor de domótica | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Una vez elegida la acción, se ejecutará una petición HTTP hacia el servidor de domótica. | | |

**Tabla 3-32: Requisito Software – REQ_SW_19**

| | | | |
|---|---|---|---|
| **REQ_SW_19** | Respuesta del servidor de domótica | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La respuesta del servidor de domótica se procesará para conocer si la acción se ha | | |

| | |
|---|---|
| | ejecutado con éxito o no. |

**Tabla 3-33: Requisito Software – REQ_SW_20**

| | | | |
|---|---|---|---|
| **REQ_SW_20** | Resultado de la acción ejecutada | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Se mostrará un mensaje al usuario sobre el resultado de la acción ejecutada. | | |

**Tabla 3-34: Requisito Software – REQ_SW_21**

| | | | |
|---|---|---|---|
| **REQ_SW_21** | Resultado de la acción ejecutada por pop-up | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Se mostrará un pop-up con el resultado de la acción | | |

**Tabla 3-35: Requisito Software – REQ_SW_22**

| | | | |
|---|---|---|---|
| **REQ_SW_22** | Resultado de la acción por voz | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La aplicación dictará el resultado de la acción al usuario mediante *Text To Speech*. | | |

**Tabla 3-36: Requisito Software – REQ_SW_23**

| | | | |
|---|---|---|---|
| **REQ_SW_23** | Marcador candidato | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Un marcador candidato (cuando no ha sido seleccionado todavía) se remarcará mediante un círculo rojo | | |

**Tabla 3-37: Requisito Software – REQ_SW_24**

| | | | |
|---|---|---|---|
| **REQ_SW_24** | Texto del marcador candidato | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |

| | |
|---|---|
| **Descripción** | Se mostrará el marcador candidato en la esquina superior izquierda con el siguiente texto: Procesando dispositivoXXX en X segundos. |

**Tabla 3-38: Requisito Software – REQ_SW_25**

| | | | |
|---|---|---|---|
| **REQ_SW_25** | Marcador seleccionado | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Un marcador seleccionado se remarcará mediante un círculo verde. | | |

**Tabla 3-39: Requisito Software – REQ_SW_26**

| | | | |
|---|---|---|---|
| **REQ_SW_26** | Texto del marcador seleccionado | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Se mostrará el marcador seleccionado en la esquina superior izquierda con el siguiente texto: Procesando dispositivoXXX. | | |

**Tabla 3-40: Requisito Software – REQ_SW_27**

| | | | |
|---|---|---|---|
| **REQ_SW_27** | Icono del proceso de detección | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Se mostrará un icono en la parte superior derecha si la aplicación está detectando un marcador válido. | | |

**Tabla 3-41: Requisito Software – REQ_SW_28**

| | | | |
|---|---|---|---|
| **REQ_SW_28** | Icono del proceso de habla | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Se mostrará un icono en la parte superior derecha si la aplicación está hablando al usuario mediante *Text To Speech*. | | |

**Tabla 3-42: Requisito Software – REQ_SW_29**

| | | | |
|---|---|---|---|
| **REQ_SW_29** | Icono del proceso de reconocimiento de voz | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Se mostrará un icono en la parte superior derecha si la aplicación está reconociendo la voz del usuario. | | |

**Tabla 3-43: Requisito Software – REQ_SW_30**

| | | | |
|---|---|---|---|
| **REQ_SW_30** | Icono del proceso de petición al servidor de domótica | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | Se mostrará un icono en la parte superior derecha si la aplicación está ejecutando una petición HTTP al servidor de domótica. | | |

**Tabla 3-44: Requisito Software – REQ_SW_31**

| | | | |
|---|---|---|---|
| **REQ_SW_31** | Menú de configuración | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | Alta |
| **Descripción** | La aplicación tendrá un menú de configuración | | |

**Tabla 3-45: Requisito Software – REQ_SW_32**

| | | | |
|---|---|---|---|
| **REQ_SW_32** | Configuración del tipo de selector de acciones | | |
| **Prioridad** | Alta | **Necesidad** | Esencial |
| **Estabilidad** | Estable | **Verificabilidad** | |
| **Descripción** | El tipo de interfaz para elegir la acción y mostrar su resultado se podrá configurar entre las opciones de texto, TTS y reconocimiento de voz a través del menú de opciones | | |

**Tabla 3-46: Requisito Software – REQ_SW_33**

| | | | |
|---|---|---|---|
| **REQ_SW_33** | Configuración del tiempo del marcador | | |
| **Prioridad** | Baja | **Necesidad** | Deseable |

| | | | |
|---|---|---|---|
| **Estabilidad** | Inestable | **Verificabilidad** | Alta |
| **Descripción** | El tiempo necesario para fijar la cámara y ejecutar la acción se podrá seleccionar desde el menú. El valor será de 1 a 5 segundos. | | |

**Tabla 3-47: Requisito Software – REQ_SW_34**

| | | | |
|---|---|---|---|
| **REQ_SW_34** | Configuración del servidor de domótica | | |
| **Prioridad** | Baja | **Necesidad** | Deseable |
| **Estabilidad** | Inestable | **Verificabilidad** | Alta |
| **Descripción** | La dirección IP y el puerto de conexión se podrán configurar desde el menú de opciones. | | |

**Tabla 3-48: Requisito Software – REQ_SW_35**

| | | | |
|---|---|---|---|
| **REQ_SW_35** | Fichero interno de configuración | | |
| **Prioridad** | Media | **Necesidad** | Deseable |
| **Estabilidad** | | **Verificabilidad** | Alta |
| **Descripción** | La aplicación cargará un fichero interno de configuración con el listado de dispositivos de la casa domótica y las posibles acciones a | | |

## 3.4 MATRIZ DE TRAZABILIDAD DE REQUISITOS

La siguiente tabla muestra la trazabilidad entre los requisitos de usuario y software, es decir como se consiguen los requisitos de usuario a través de los requisitos software.

**Tabla 3-49: Matriz de Trazabilidad Requisitos Usuario – Requisitos Software**

| | REQ_USER_01 | REQ_USER_02 | REQ_USER_03 | REQ_USER_04 | REQ_USER_05 | REQ_USER_06 | REQ_USER_07 | REQ_USER_08 | REQ_USER_09 | REQ_USER_10 | REQ_USER_11 | REQ_USER_12 | REQ_USER_13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **REQ_SW_01** | X | | | | | | | | | | | | |
| **REQ_SW_02** | | X | | | | | | | | | | | |
| **REQ_SW_03** | | X | | | | | | | | | | | |
| **REQ_SW_04** | | X | | | | | | | | | | | |
| **REQ_SW_05** | | X | | | | | | | | | | | |
| **REQ_SW_06** | | | X | X | | | | | | | | | |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **REQ_SW_07** | | | X | X | | | | | | | | | |
| **REQ_SW_08** | | | | | X | | | | | | | | |
| **REQ_SW_09** | | X | | | | | | | | | | | |
| **REQ_SW_10** | | | | X | | | | | | | | | |
| **REQ_SW_11** | | | | | | | X | X | | | | | |
| **REQ_SW_12** | | | | | | | | X | | | | | |
| **REQ_SW_13** | | | | | | | X | | | | | | |
| **REQ_SW_14** | | | | | | | | | X | X | | | |
| **REQ_SW_15** | | | | | | | | | | X | | | |
| **REQ_SW_16** | | | | | | | | | X | | | | |
| **REQ_SW_17** | | | | X | | | | | | | | | |
| **REQ_SW_18** | | | | X | | | | | | | | | |
| **REQ_SW_19** | | | | X | | | | | | | | | |
| **REQ_SW_20** | | | | | | | X | X | | | | | |
| **REQ_SW_21** | | | | | | | X | | | | | | |
| **REQ_SW_22** | | | | | | | | X | | | | | |
| **REQ_SW_23** | | | | | | X | | | | | | | |
| **REQ_SW_24** | | | | | | X | | | | | | | |
| **REQ_SW_25** | | | | | | X | | | | | | | |
| **REQ_SW_26** | | | | | | X | | | | | | | |
| **REQ_SW_27** | | X | | | | | | | | | | | |
| **REQ_SW_28** | | | | | | | X | | | | | | |
| **REQ_SW_29** | | | | | | | | | X | | | | |
| **REQ_SW_30** | | | | | | X | | | | | | | |
| **REQ_SW_31** | | | | | | | | | | | X | X | X |
| **REQ_SW_32** | | | | | | | | | | | X | | |
| **REQ_SW_33** | | | | | | | | | | | | | X |
| **REQ_SW_34** | | | | | | | | | | | | X | |
| **REQ_SW_35** | | | X | X | | | | | | | | | |

## 3.5 DIAGRAMAS DE ACTIVIDAD

Los siguientes diagramas de actividad muestran un primer análisis del flujo de información. En primer lugar cómo se reconoce un marcador en el sistema.

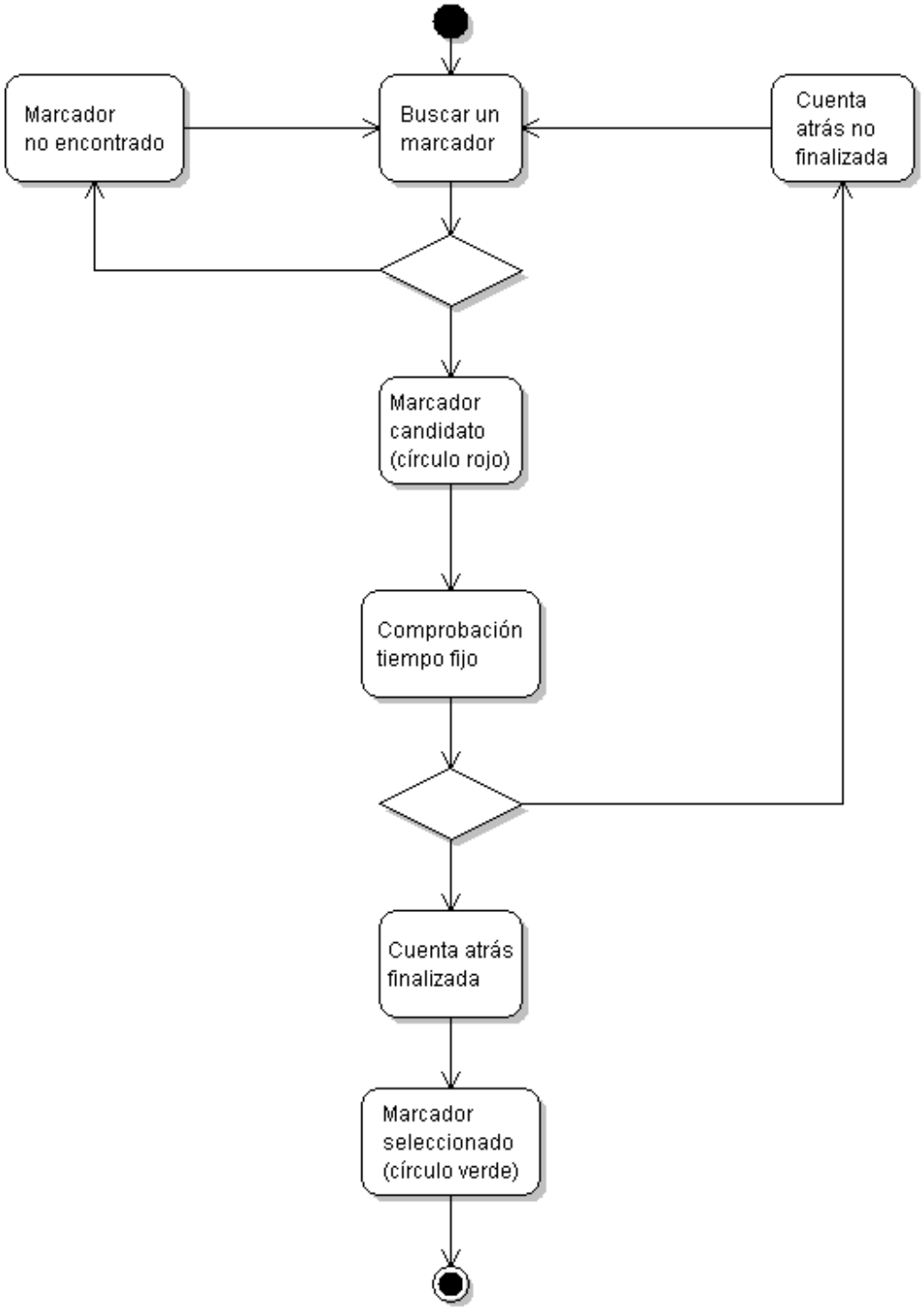


**Figura 3-5: Diagrama de Actividad – Buscar un Marcador**

A continuación como se muestra una acción al usuario ya sea por texto o por TTS.

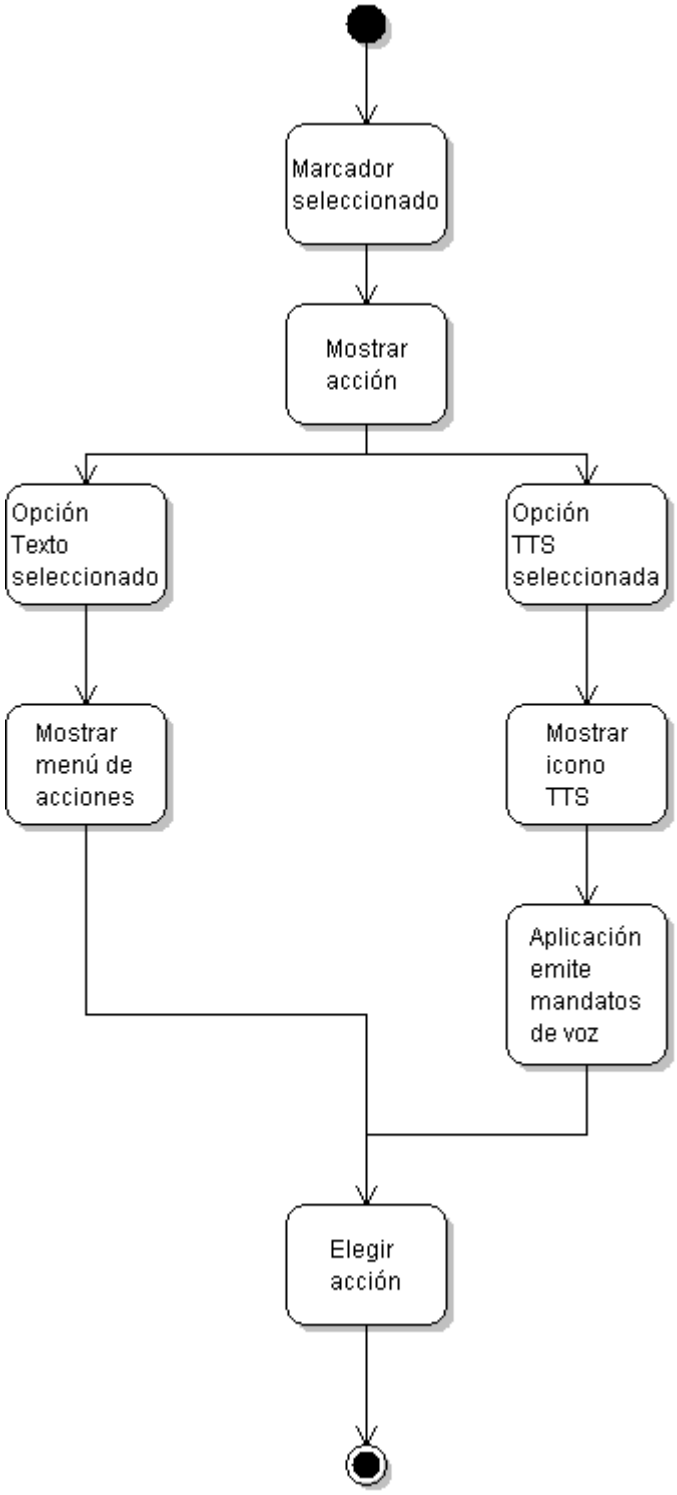


**Figura 3-6: Diagrama de Actividad –Marcador Seleccionado**

Este diagrama muestra cómo se elige una acción en el sistema, mediante la interfaz de texto o mediante reconocimiento de voz.

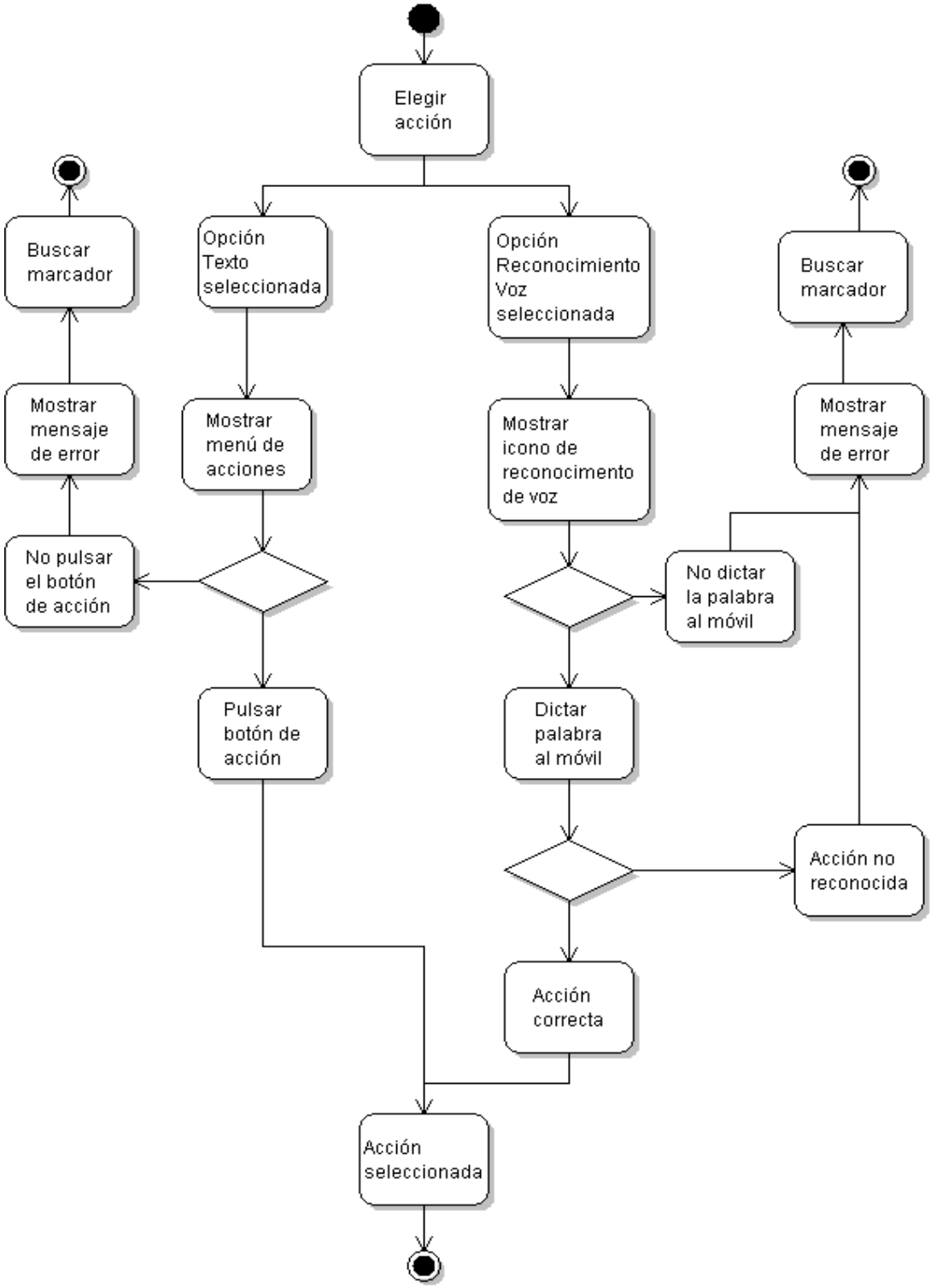


**Figura 3-7: Diagrama de Actividad – Elegir una Acción**

Por último se detalla cómo se ejecuta una acción en el servidor de domótica y se muestra el resultado al usuario.

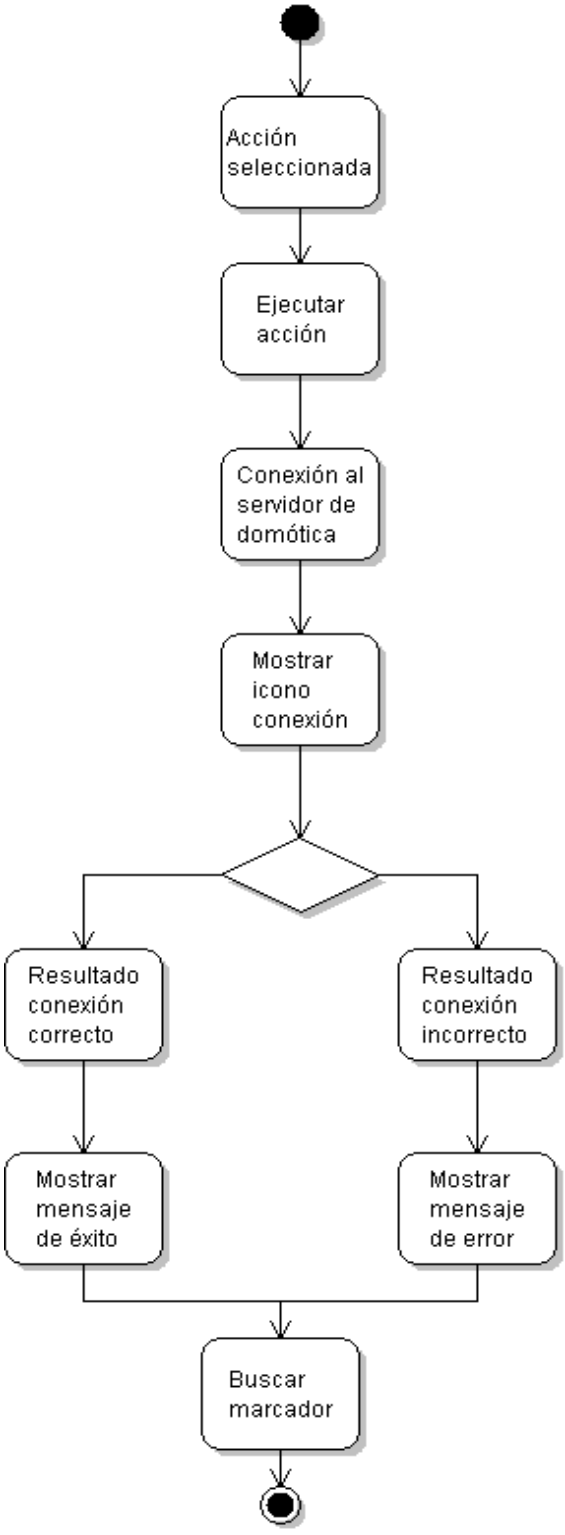


**Figura 3-8: Diagrama de Actividad –Acción Seleccionada**

# 4. DISEÑO

En esta sección se describirá mediante el uso de diagramas UML el diseño de la aplicación Android, a través de un primer diagrama de componentes de alto nivel, para continuar con el diagrama de clases y los diagramas de secuencia y finalizar con los diagramas de navegación que representan la interfaz de usuario del sistema.

## 4.1 DIAGRAMA DE COMPONENTES

El diagrama de componentes de la aplicación es el definido por la siguiente figura, en la que se muestran los distintos paquetes o módulos de la aplicación.

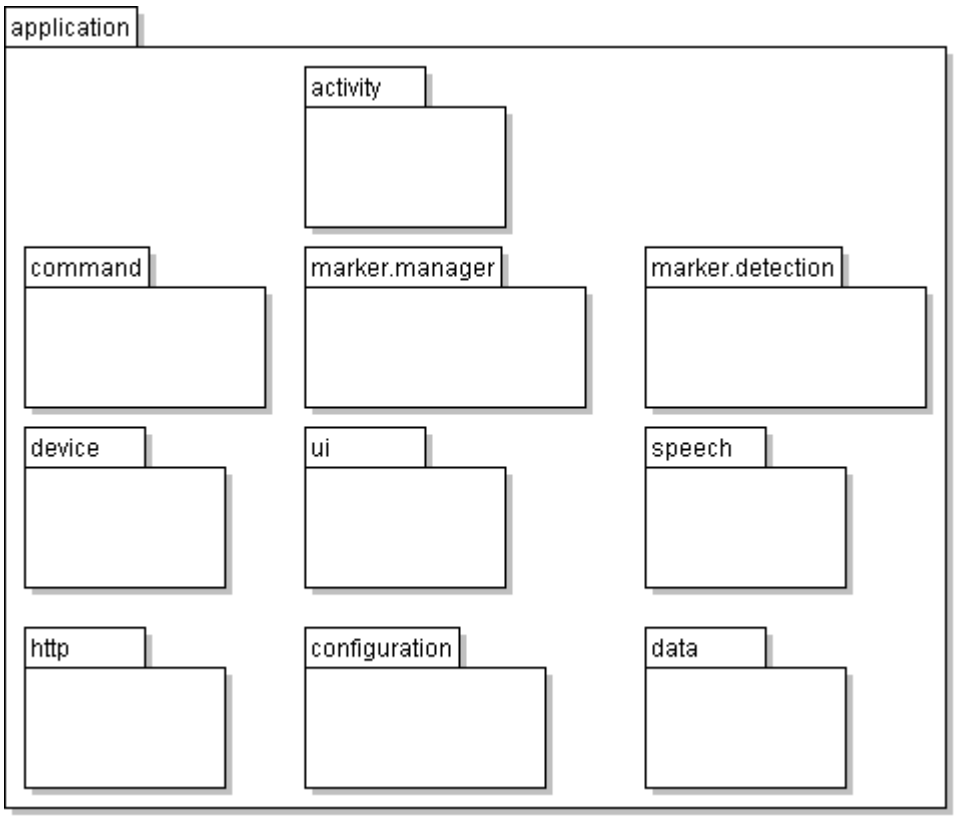


**Figura 4-1: Diagrama de Componentes**

Una primera aproximación a estos módulos es la descrita a continuación:

- Módulo *activity*: encargado de la implementación de la actividad principal de una aplicación android y los distintos componentes de la interfaz gráfica.
- Módulo *marker*: responsable de la gestión de los marcadores. Se puede dividir en:

  - Módulo *detection*: detecta marcadores capturados por la cámara de la aplicación.
  - Módulo *manager*: gestiona los marcadores encontrados y su posterior procesamiento.

- Módulo *command*: encargado de iniciar la ejecución de una acción hacia el dispositivo domótico una vez seleccionado un marcador.
- Módulo *device:* responsable de la asociación entre marcadores y dispositivos/acciones y de realizar las peticiones de conexión al servidor de domótica.
- Módulo *ui:* gestiona la interacción con la actividad principal del sistema para notificaciones al usuario.
- Módulo *speech:* agrupa el tratamiento de la tecnología TTS y el reconocimiento de voz en la aplicación.
- Módulo *http:* realiza peticiones HTTP hacia el servidor de domótica para ejecutar una acción sobre un dispositivo.
- Módulo *configuration:* encargado de leer el fichero de configuración en el que se definen los dispositivos existentes en el sistema domótico y las posibles acciones a realizar.
- Módulo *data:* contiene el modelado de clases del sistema domótico que se cargará mediante el fichero de configuración.

## 4.2 DIAGRAMA DE CLASES

El siguiente diagrama de clases muestra la aplicación a un alto nivel, dividiéndose las clases por el módulo o paquete que lo contiene.

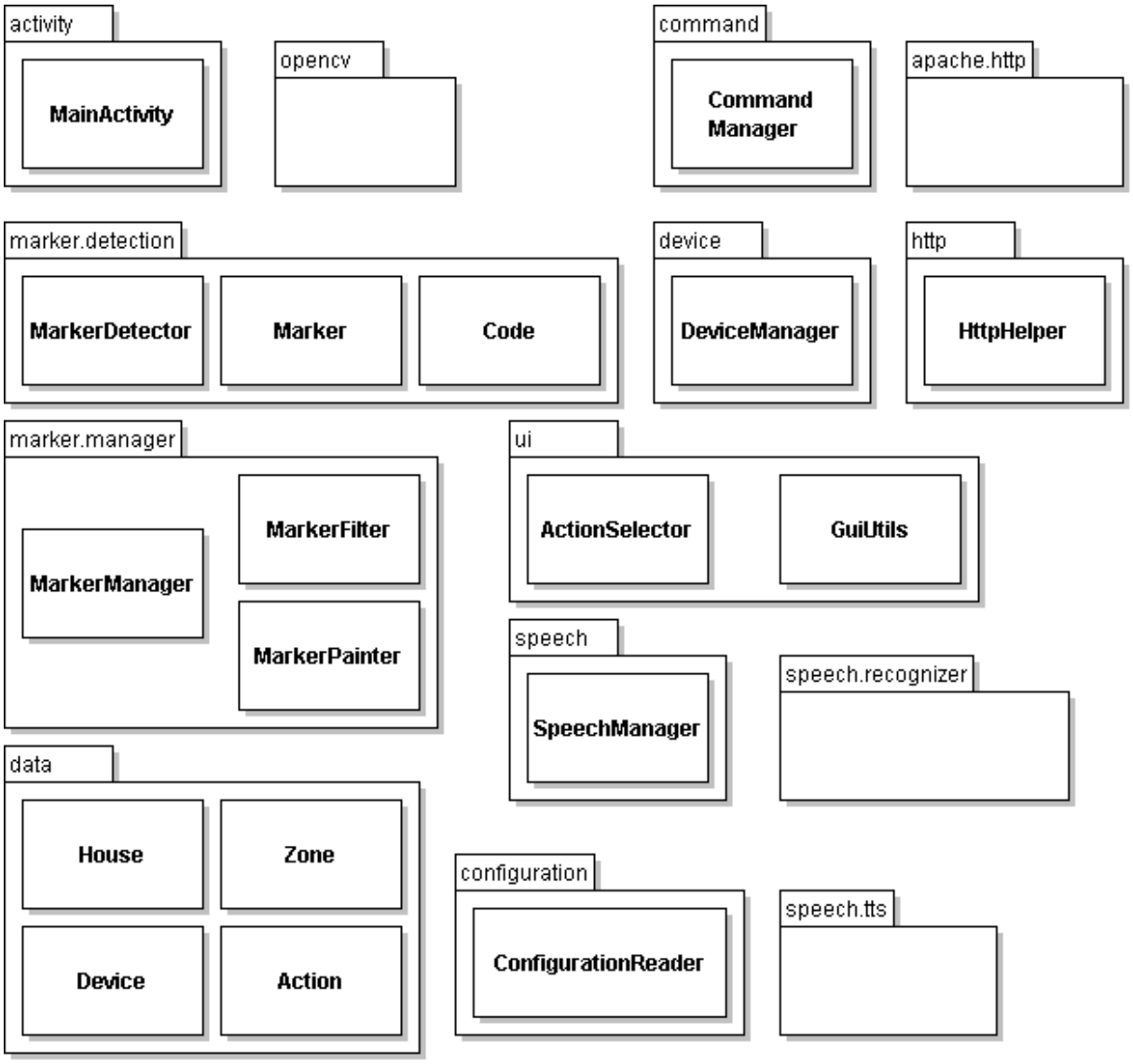


**Figura 4-2: Diagrama de Clases**

Una descripción de la funcionalidad de las clases es la siguiente:

- Módulo *activity:*
  - Clase *MainActivity*: muestra al usuario los *frames* recibidos por la captura de video para su posterior tratamiento y permite la interacción del usuario para ejecutar acciones. Utiliza el API externo:
    - Librería *OpenCV*: en este caso es utilizado para la captura de video y

recepción de cada trama por el sistema.

- Módulo *marker:*
  - Módulo *detection*:
    - Clase *MarkerDetector*: a partir del frame con la imagen recibido por la aplicación implementa el algoritmo de detección de marcadores y la extracción del código entero almacenado en el marcador. Utiliza el API:
      - Librería *OpenCV*: en este caso es utilizado para ejecutar procesado de imagen sobre el frame capturado para identificar el marcador.
    - Clase *Marker*: objeto que encapsula un marcador físico en el sistema.
    - Clase *Code*: código asociado al marcador capturado.
  - Módulo *manager*:
    - Clase *MarkerManager*: gestiona el estado del marcador en el sistema, si es candidato o ya ha sido seleccionado para su posterior procesamiento.
    - Clase *MarkerFilter*: encargado de filtrar marcadores y así habilitar o no su procesado. Diferencia entre filtro por tiempo mínimo de reconocimiento o filtro por marcadores múltiples.
    - Clase *MarkerPainter*: modifica el frame original para añadir la información necesaria para el sistema como texto del marcador o el círculo sobre el marcador reconocido.
- Módulo *command:*
  - Clase *CommandManager*: procesa la ejecución de una acción sobre el dispositivo y el flujo de información una vez el marcador ha sido seleccionado por el usuario.
- Módulo *device:*
  - Clase *DeviceManager*: identifica marcadores contra dispositivos y acciones domóticas, así como la interacción contra el servidor de domótica.
- Módulo *ui:*
  - Clase *ActionSelector*: encargado de la selección de una acción en el sistema, ya sea por texto, tts y/o reconocimiento de voz.
  - Clase *GuiUtils*: permite mostrar el menú de acciones al usuario así como mensajes de información. Además posibilita la ejecución de procesos en el hilo de la interfaz gráfica.
- Módulo *speech*
  - Clase *SpeechManager*: permite emitir comandos de voz y/o reconocer voz para la selección de una acción en el sistema. Utiliza los APIs externos:
    - Librería *SpeechRecognizer*: implementa el servicio de reconocimiento

de voz en android.

        - Librería *TTS*: implementa el *engine* de Text To Speech en android.

- Módulo *http:*
    - Clase *HttpHelper*: realiza conexiones HTTP Get hacia el servidor de domótica procesando los posibles fallos de comunicaciones. Utiliza el API externo:
        - Librería *ApacheHTTP*: implementa el protocolo de comunicación HTTP sobre android.
- Módulo *configuration:*
    - Clase *ConfigurationReader*: lee el fichero de configuración interno en formato JSON con el listado de dispositivos domóticos y las posibles acciones a realizar.
- Módulo *data:*
    - Clase *House*: modela los datos de una casa domótica.
    - Clase *Zone*: modela los datos de una zona de la casa domótica.
    - Clase *Device*: modela los datos de un dispositivo de la casa domótica.
    - Clase *Action*: modela los datos de una acción de la casa domótica.

La siguiente imagen muestra el diagrama de contexto con las interacciones entre las distintas clases del sistema, que se detallarán más a fondo en el siguiente apartado.

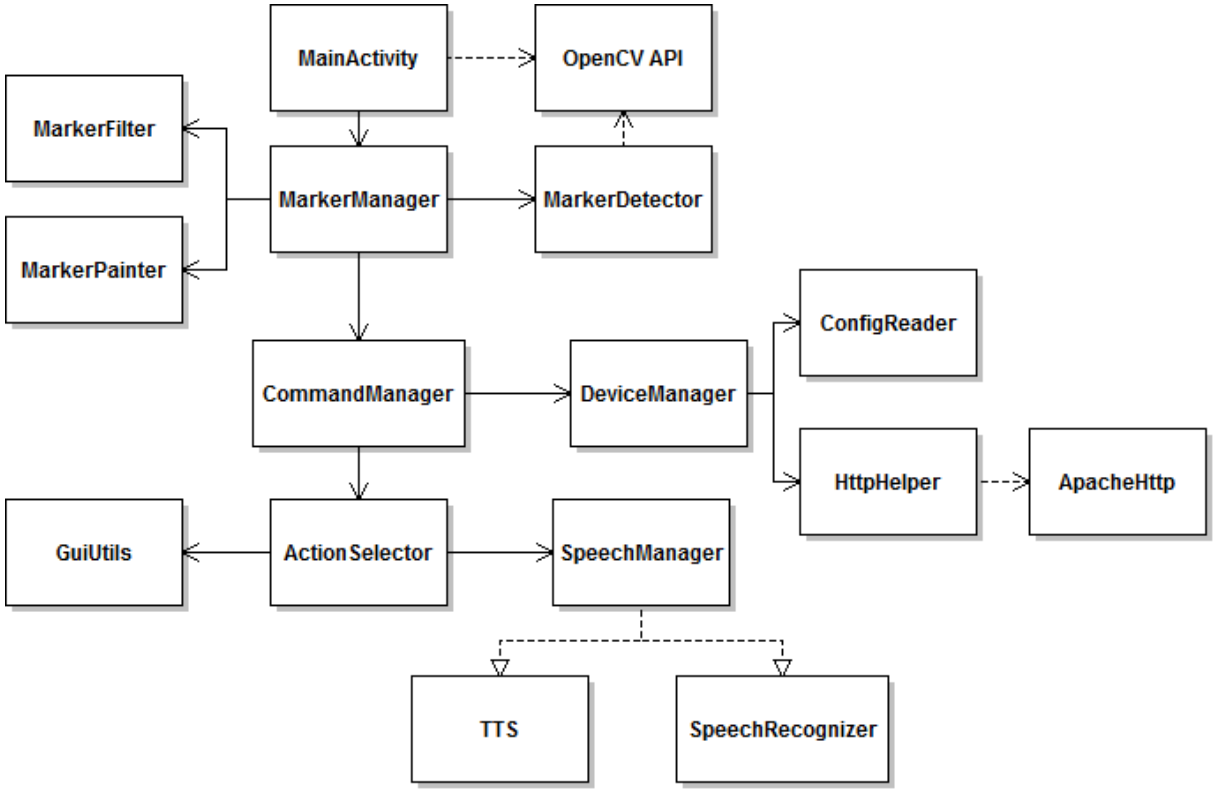


**Figura 4-3: Diagrama de Contexto**

## 4.3 DIAGRAMAS DE SECUENCIA

Los diagramas de secuencia de este apartado modelan las siguientes interacciones:

- Gestión de un marcador.
- Ejecución de la acción.
- Selección de la acción.
- Detección de un marcador.

### 4.3.1 Gestión de un Marcador

El siguiente diagrama muestra la llegada de una nueva imagen al sistema, la detección y filtrado de marcadores, así como la actualización de la información del marcador en pantalla y la ejecución del comando.

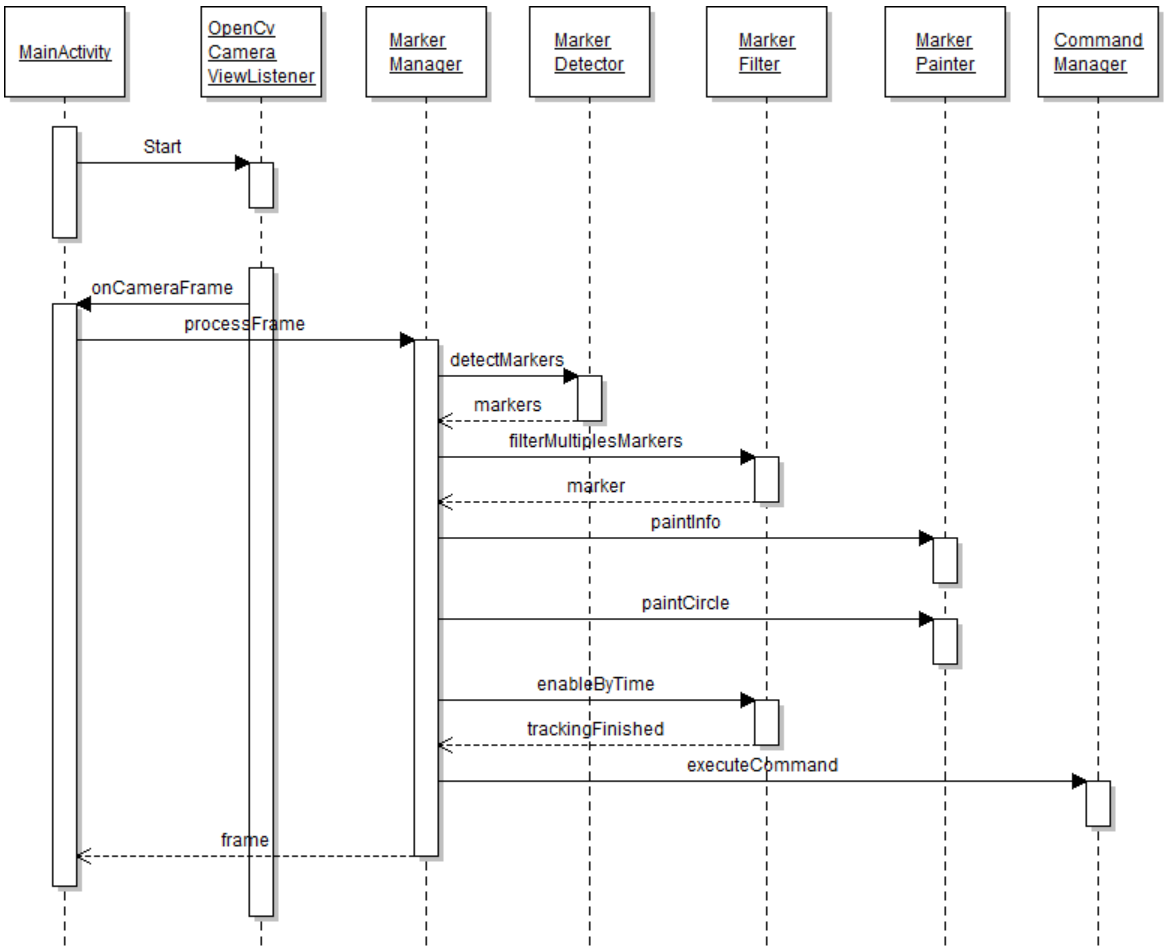


**Figura 4-4: Diagrama de Secuencia – Gestión de un Marcador**

### 4.3.2 Ejecución de la Acción

El siguiente diagrama muestra la obtención del dispositivo y sus acciones mediante el marcador seleccionado, la selección de la acción por parte del usuario y la petición HTTP al servidor de domótica.

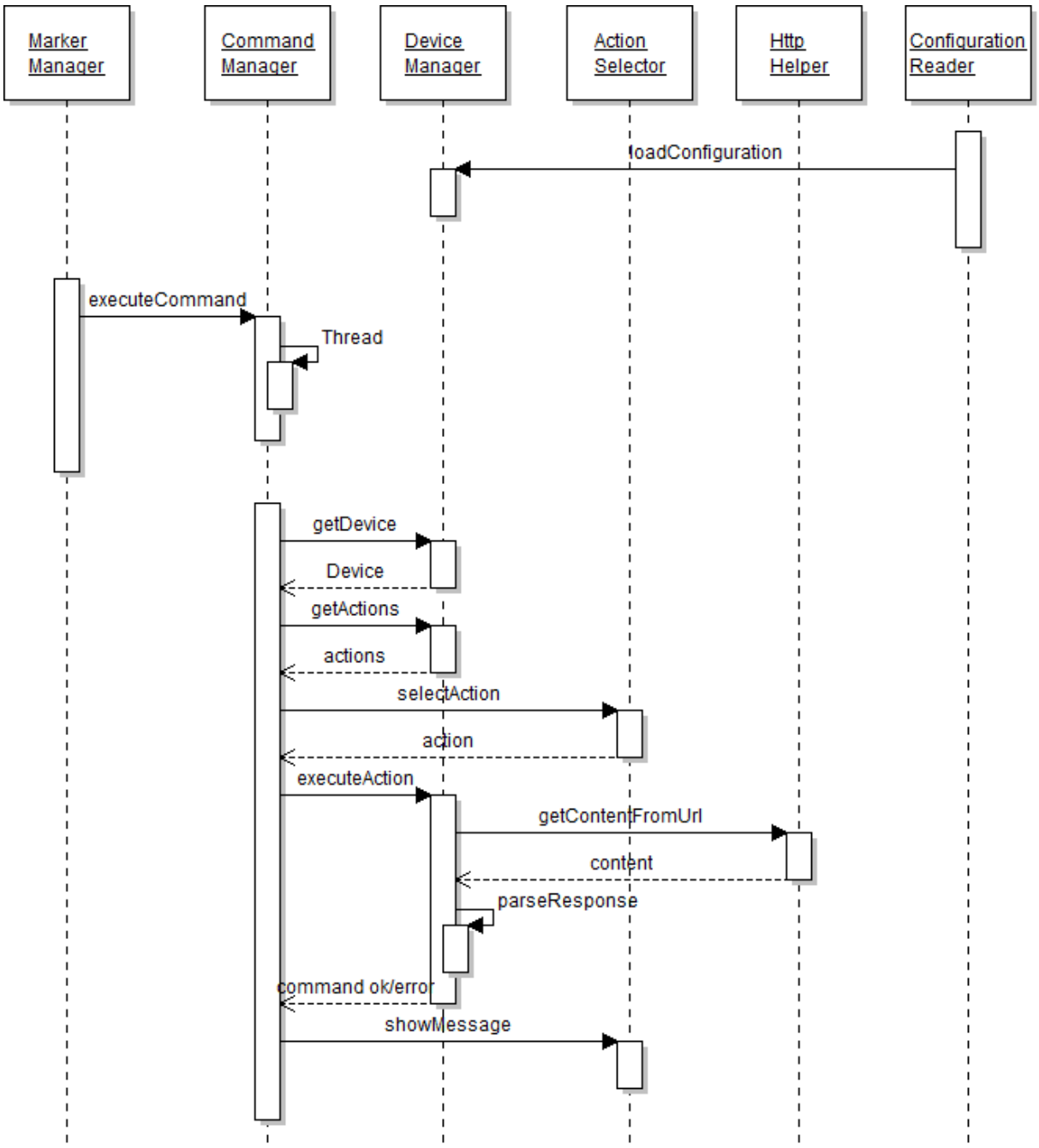


**Figura 4-5: Diagrama de Secuencia – Ejecución de la Acción**

### 4.3.3 Selección de la Acción

El diagrama mostrado a continuación describe la selección de la acción por tipo de interfaz, ya sea texto, TTS y/o reconocimiento de voz.

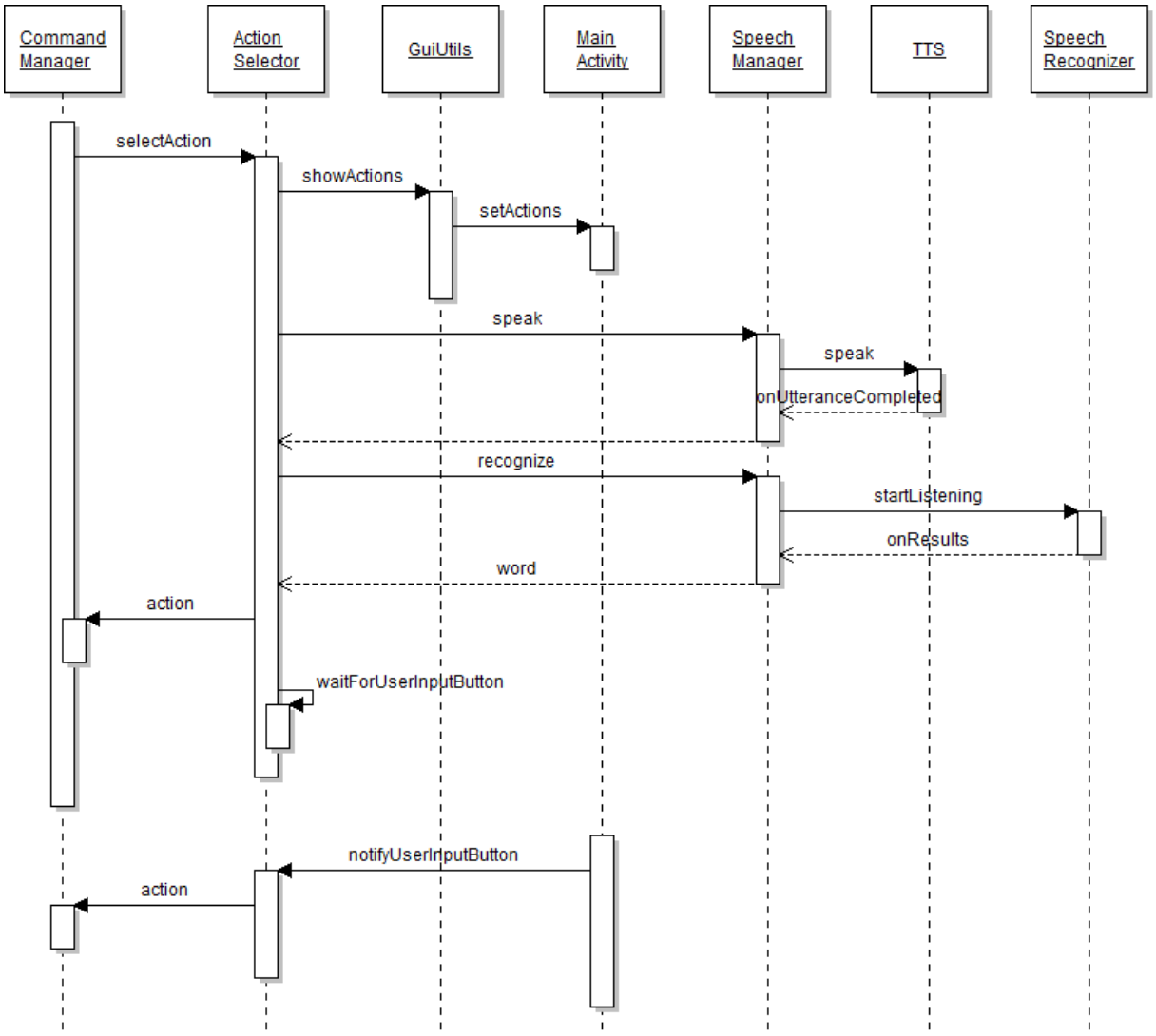


**Figura 4-6: Diagrama de Secuencia – Selección de una Acción**

### 4.3.4 Detección de un Marcador

La siguiente figura muestra las operaciones realizadas para detectar un marcador en el sistema. En primer lugar se modifica el frame o imagen original pasándolo a escala de grises y aplicando un *threshold* para diferenciar los saltos de color. Con este frame modificado se buscan los contornos y para cada contorno se comprueba si es un cuadrado y se crea el marcador.

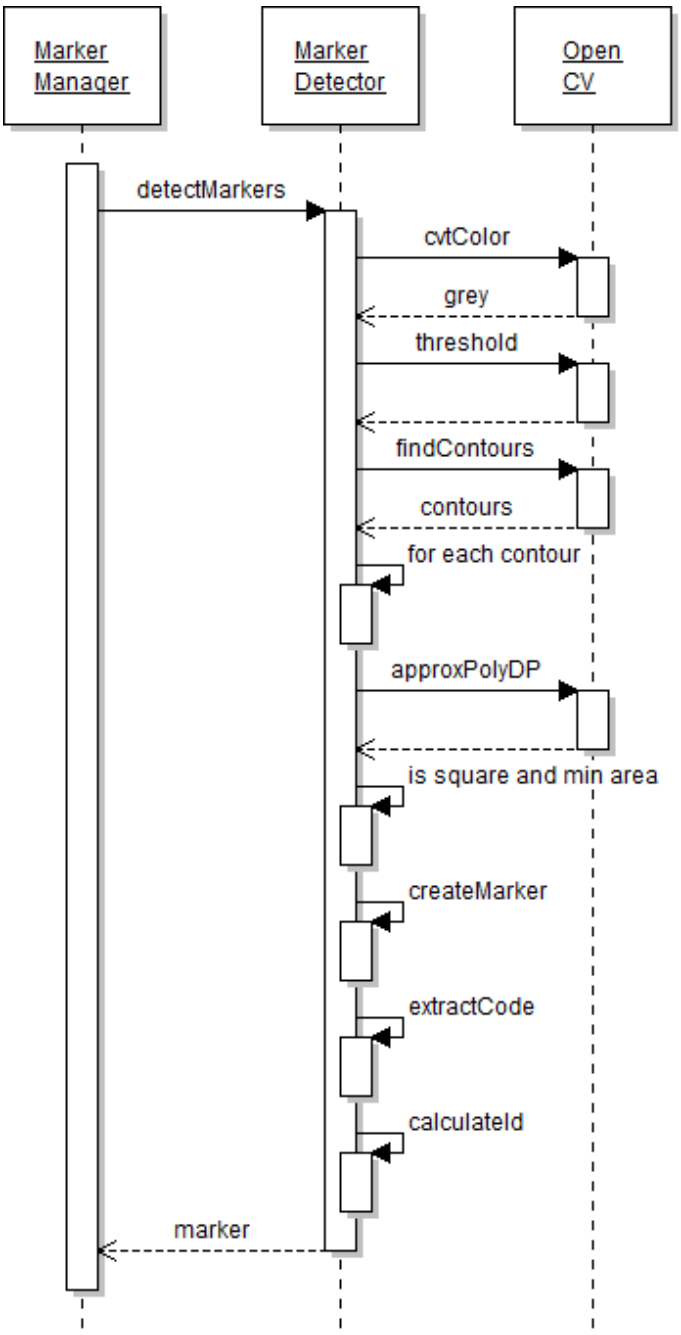


**Figura 4-7: Diagrama de Secuencia - Detección de un Marcador**

## 4.4 DIAGRAMA DE DESPLIEGUE

El diagrama de despliegue de la aplicación es bastante sencillo en nuestro caso, ya que solo se trata de un usuario con un dispositivo móvil smartphone, con conexión a internet habilitada, para la realización de peticiones HTTP al servidor de domótica.


**Figura 4-8: Diagrama de Despliegue**

## 4.5 DIAGRAMAS DE NAVEGACIÓN

Los siguientes diagramas muestran el flujo de navegación por la interfaz de usuario.

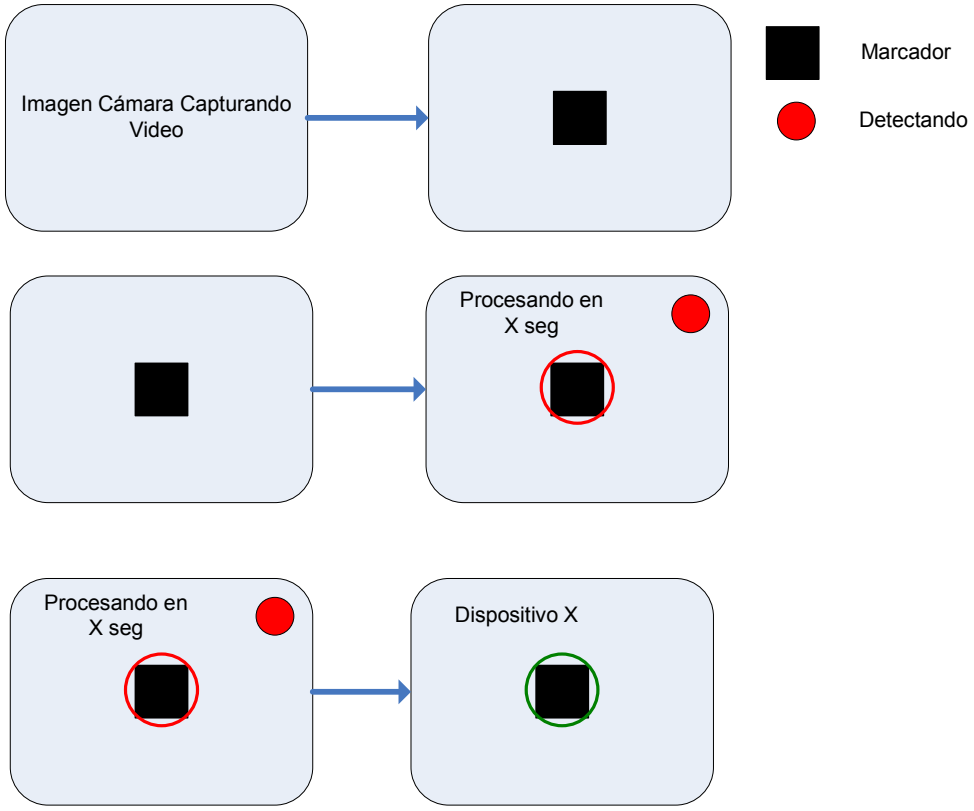


**Figura 4-9: Diagrama de Navegación – Reconocimiento de un Marcador**

1. Con la aplicación iniciada se busca un marcador.
2. Al reconocer un marcador se establece en primer lugar como candidato, con un círculo en rojo y el icono de detección. Se muestra el texto de información “Procesando en X segundos”.
3. Pasado el tiempo de reconocimiento mínimo el marcador ya está seleccionado, con un círculo en verde. Se muestra el texto de información “Dispositivo X”

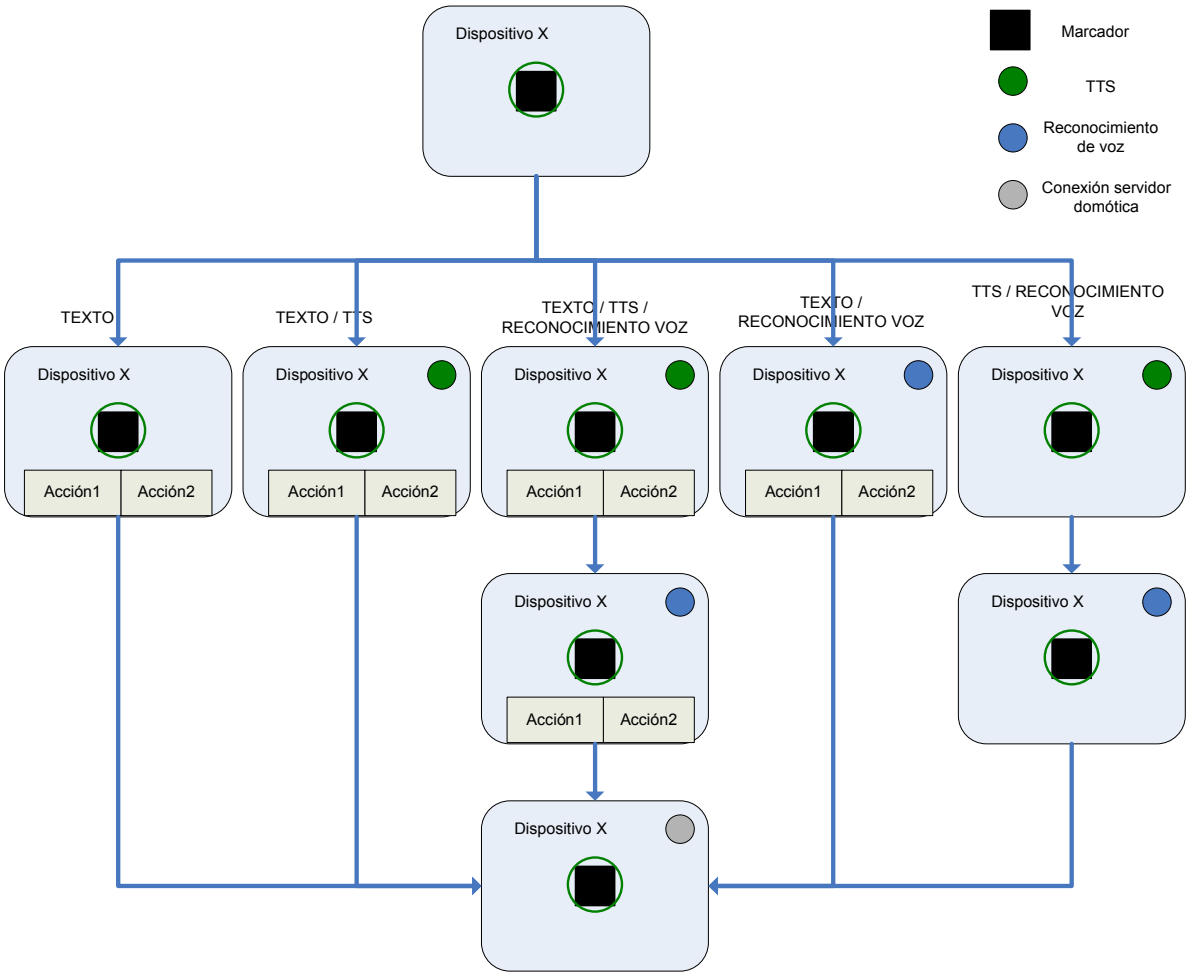


**Figura 4-10: Diagrama de Navegación – Selección de la Acción**

4. Interfaz modo Texto: se muestra el listado de acciones a través del menu de opciones.
5. Interfaz modo Texto/TTS: se muestra el listado de acciones a través del menu de opciones y también se emite el listado de acciones por voz. Se muestra el icono de TTS.
6. Interfaz modo Texto/Reconocimiento de voz: se muestra el listado de acciones a través del menu de opciones y también se espera la selección de la acción por reconocimiento de voz. Se muestra el icono de reconocimiento de voz.
7. Interfaz TTS: se emite el listado de acciones por voz. Se muestra el icono de TTS.
8. Interfaz de reconocimiento de voz: se espera la selección de la acción por reconocimiento de voz. Se muestra el icono de reconocimiento de voz.
9. Conexión HTTP al servidor de domótica: se realiza la conexión al servidor de domótica. Se muestra el icono de conexión.

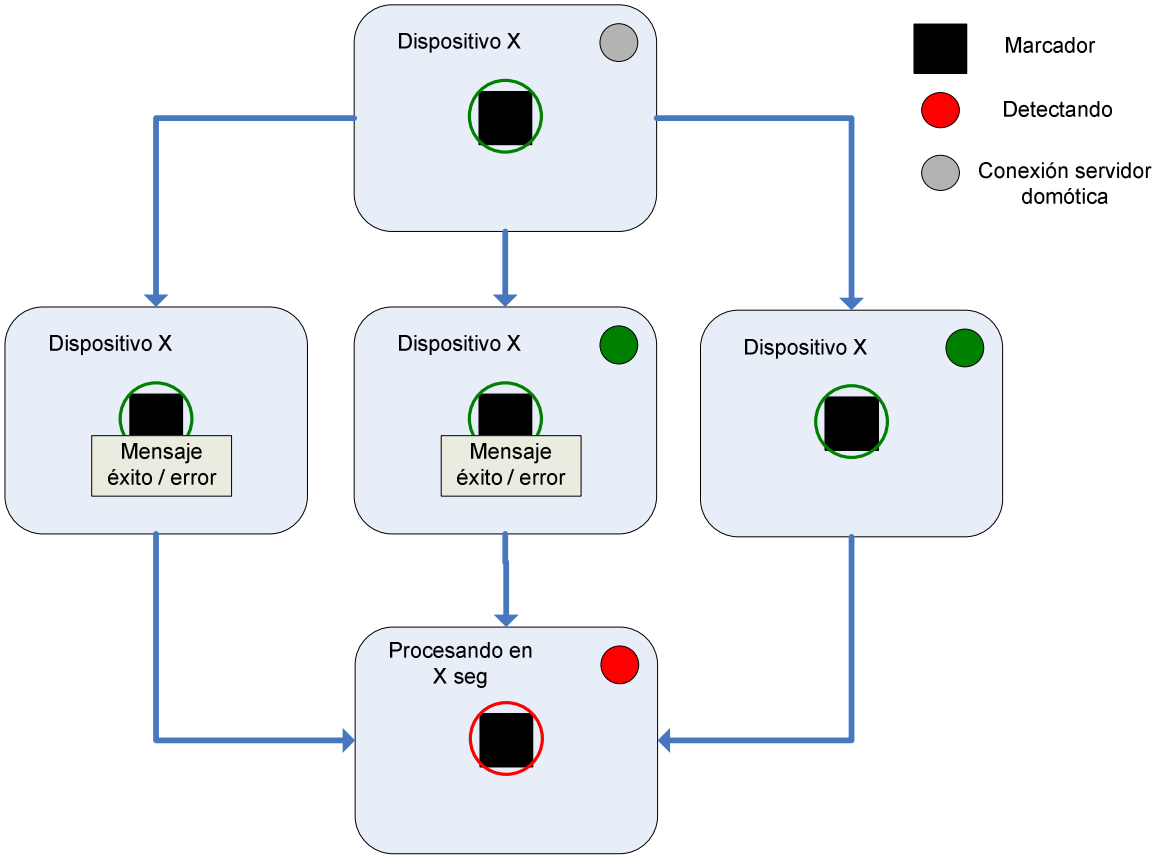


**Figura 4-11: Diagrama de Navegación – Resultado de la Acción**

10. Mensaje de información por pantalla: se muestra un pop-up con el mensaje de éxito o de error al realizar la conexión.
11. Mensaje de información por pantalla y por voz: se muestra un pop-up con el mensaje de éxito o de error al realizar la conexión. Además se emite el mismo mensaje por voz.
12. Mensaje de información por voz: se emite el mensaje de éxito o de error por voz.

# 5. IMPLEMENTACIÓN

En esta sección se describirá tanto el entorno de desarrollo utilizado para la realización del proyecto, como las librerías externas utilizadas, mostrando su uso dentro del código fuente del proyecto.

## 5.1 ENTORNO DE DESARROLLO

### 5.1.1 Eclipse

Android Developer Tools: Build: v21.1.0-569685

### 5.1.2 Android SDK

Android 4.2.2 (API 17)

## 5.2 UTILIZACIÓN DE APIS

### 5.2.1 Open CV

La versión de la librería de OpenCV para Android utilizada ha sido: OpenCV-2.4.5. A continuación se mostrará la utilización de está librería en las clases más importantes de la aplicación:

#### 5.2.1.1 Main Activity

- Implementa el listener para obtener los frames de la cámara.

```
public final class MainActivity extends Activity implements
CvCameraViewListener2
```

- Establece la vista base de la actividad.

```
    @Override
    public void onCreate(final Bundle savedInstanceState) {
            mOpenCvCameraView = (CameraBridgeViewBase)
findViewById(R.id.tutorial1_activity_java_surface_view);
```

- Inicia la carga asíncrona de la librería.

```
@Override
public void onResume() {
    super.onResume();
    OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_5, this,
            mLoaderCallback);
}
```

- Procesa cada frame de la cámara recibido en la aplicación para su procesamiento.

```
@Override
public Mat onCameraFrame(final CvCameraViewFrame inputFrame) {
 markerManager.processFrame(inputFrame.rgba());
```

### 5.2.1.2 Marker Detector

- Modifica el frame a la escala de grises.
- Implementa un threshold inverso sobre el frame modificado.
- Encuentra los contornos de dicho frame una vez aplicado el threshold.

```
public void process(final Mat in) {

Imgproc.cvtColor(in, grey, Imgproc.COLOR_RGBA2GRAY);

Imgproc.threshold(grey, dst, MIN_THRESHOLD, MAX_THRESHOLD,
                Imgproc.THRESH_BINARY_INV);

Imgproc.findContours(dst, contours, hierarchy, Imgproc.RETR_LIST,
                Imgproc.CHAIN_APPROX_NONE);
```

- Para cada contorno encontrado en el punto anterior:
  - Comprueba si el área es mayor que el mínimo.
  - Se aproxima el contorno a un polígono.
  - Si el polígono es un cuadrado, esa curva/contorno puede ser un marcador.
  - Se crea un marcador a partir del contorno
  - Se detecta el marcador, extrayendo el código entero y comprobando su validez.

```
for (MatOfPoint contourOriginal : contours) {
     if (Math.abs(Imgproc.contourArea(contourOriginal)) > MIN_AREA) {

     Imgproc.approxPolyDP(contour, approxCurve,
          Imgproc.arcLength(contour, true) * DELTA_FOR_ARC, true);

     square = isSquare(approxCurve, convex);

     Marker marker = createMarker(approxCurve, contourOriginal);
     detectMarker(marker, in);
```

- Para crear un marcador:
  - Se obtiene los puntos de esa curva.
  - Se crea el listado de 4 puntos que contienen los vértices del cuadrado.

```
private Marker createMarker(final MatOfPoint2f papproxCurve,
               final MatOfPoint contourOriginal) {

          papproxCurve.get(0, 0, points);

          int i = -1;
          List<Point> p = new ArrayList<Point>();
          p.add(new Point(points[++i], points[++i]));
          p.add(new Point(points[++i], points[++i]));
          p.add(new Point(points[++i], points[++i]));
          p.add(new Point(points[++i], points[++i]));

          Marker marker = new Marker(MARKER_SIZE, p);
          marker.setContour(contourOriginal);
```

- Para detectar un marcador:
  - Se empaqueta el listado de puntos de ese marcador en un marcador canónico.
  - Se extrae el código entero del nuevo marcador.
  - Si es un marcador válido (distinto de -1 y de 0) se añade si no existe ya en la lista de nuevos marcadores reconocidos.

```
private void detectMarker(final Marker marker, final Mat in) {

     warp(in, canonicalMarker, canonicalSize, marker.toList());

     marker.setMat(canonicalMarker);
     marker.extractCode();

     int id = marker.calculateMarkerId();
     if (id != -1 && id != 0) {
          if (!newMarkers.contains(marker)) {
                    newMarkers.add(marker);
          }
     }
```

### 5.2.2 Text To Speech

La funcionalidad de TextToSpeech es provista por defecto por el API de Android a través del paquete "android.speech.tts". Su uso se ha encapsulado en la clase *SpeechManager* que maneja tanto el TTS como el reconocimiento de voz.

#### 5.2.2.1 Speech Manager

- Implementa la interfaz:
  - TextToSpeech.OnInitListener: para iniciar el componente tts.
  - TextToSpeech.OnUtteranceCompletedListener: para conocer que el *speech* ha finalizado.

```
public final class SpeechManager implements
     TextToSpeech.OnInitListener,
     TextToSpeech.OnUtteranceCompletedListener {

     /** TTS instance. */
     private TextToSpeech tts;
```

- Para la inicialización del engine de TTS es necesario el contexto de la aplicación.

```
     private void start() {

          // create the TTS object
          tts = new TextToSpeech(context, this);
```

- La aplicación emite los comandos de voz invocando el método *speak* del objeto *tts* y esperando su finalización (el método *tts* no es bloqueante).

```
private void speakOut(final String text, final SpeechType speechType) {
HashMap<String, String> myHashAlarm = new HashMap<String, String>();
     myHashAlarm.put(TextToSpeech.Engine.KEY_PARAM_STREAM,
                     String.valueOf(AudioManager.STREAM_ALARM));
     myHashAlarm.put(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID,
                     speechType.name());

     tts.speak(message, TextToSpeech.QUEUE_FLUSH, myHashAlarm);
     waitForSpeech();
```

- El resultado de la inicialización del engine de TTS se notifica en esté método, dónde también se establece el idioma a utilizar (en este caso el lenguaje por defecto).

```
@Override
public void onInit(final int status) {

     if (status == TextToSpeech.SUCCESS) {

          int result = tts.setLanguage(Locale.getDefault());

          if (result == TextToSpeech.LANG_MISSING_DATA
               || result == TextToSpeech.LANG_NOT_SUPPORTED) {

               Log.e("TTS", "This Language is not supported");
          } else {
               tts.setOnUtteranceCompletedListener(this);
          }

     } else {
          Log.e("TTS", "Initilization Failed!");
     }

}
```

- Notificación de la finalización del *speech* que despierta el hilo bloqueado anteriormente.

```
@Override
public void onUtteranceCompleted(final String utteranceId) {

     onFinishSpeech(utteranceId);
}


private void onFinishSpeech(final String id) {


     notifyFromSpeech();
}
```

### 5.2.3 Voice Recognizer

La funcionalidad de reconocimiento de voz es provista por defecto por el API de Android a través del paquete “android.speech”. Su uso se ha encapsulado en la clase *SpeechManager* que maneja tanto el TTS como el reconocimiento de voz.

#### 5.2.3.1 Speech Manager

- Implementa la interfaz:
  - RecognitionListener: para implementar los métodos de reconocimiento.

```
public final class SpeechManager implements RecognitionListener
     /** Intent for voice recognition. */
     private Intent intent;
     /** Speech recognizer object. */
     private SpeechRecognizer sr;
```

- Inicialización del objecto SpeechRecognizer y del objecto Intent de reconocimiento.

```
private void start() {
     // create the speech recognizer object
     sr = SpeechRecognizer.createSpeechRecognizer(context);
     sr.setRecognitionListener(this);

     // create the intent for speech recognizer
     intent = new Intent(
          RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

     intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
          RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);

     intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
          this.getClass().getPackage().getName());

     intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS,
          MAX_VOICE_RESULTS);
```

- Método para notificar que la aplicación está lista para iniciar el reconocimiento pero todavía no ha recibido ningún dato. Comienza la cuenta atrás para deshabilitar el proceso de reconocimiento si no se ha recibido ningún dato en 5 segundos.

```
@Override
public void onReadyForSpeech(final Bundle params) {
     isSpeechRecognizerAlive = false;
     mNoSpeechCountDown.start();
}
```

- Método para notificar que la aplicación ha iniciado el reconocimiento ya que ha recibido algún dato.

```
@Override
public void onBeginningOfSpeech() {
     isSpeechRecognizerAlive = true;
}
```

- Método para notificar la obtención de resultados por parte del reconocedor de voz. Una vez finalizado la búsqueda de la palabra seleccionada se notifica que el hilo de reconocimiento despierte.

```
@Override
public void onResults(final Bundle results) {
     String str = new String();
     ArrayList<String> data = results
.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);

     boolean found = false;
     for (int i = 0; i < data.size(); i++) {
          str = data.get(i).toLowerCase();

          if (words.contains(str)) {
               found = true;
               break;
          }
     }

     notifyFromRecog();
```

- Cuenta atrás para cancelar el reconocimiento si no se ha recibido ningún dato en 5 segundos de escuha.

```
/** Count down timer for Jelly Bean work around. */
private CountDownTimer mNoSpeechCountDown =
          new CountDownTimer(TIME_OUT_FOR_VOICE_RECOG,
                    TIME_OUT_FOR_VOICE_RECOG) {
     @Override
     public void onFinish() {

          if (!isSpeechRecognizerAlive) {
               if (sr != null) {
                    sr.stopListening();
                    sr.cancel();

                    setWord(null);
                    notifyFromRecog();
               }
          }

     }

};
```

### 5.2.4 Http Request

La funcionalidad de realizar peticiones HTTP es provista por defecto por el API de Android a través del paquete “org.apache.http”. Su uso se limita a la clase HttpHelper.

#### 5.2.4.1 Http Helper

- Método que dada una URL obtiene su contenido en formato *String*.

```
public String getContentFromUrl(final String url) {
     String content = null;
     InputStream in = getInputStreamFromUrl(url);
     if (in != null) {
          content = inputStreamToString(in);
     }
     return content;
}
```

- Método que dada una URL realiza una petición HTTP Get para obtener la respuesta.

```
private InputStream getInputStreamFromUrl(final String url)
     throws Exception {
     InputStream content = null;
     HttpClient httpclient = new DefaultHttpClient();

     HttpResponse response = httpclient.execute(new HttpGet(url));
     content = response.getEntity().getContent();
     return content;
     }
```

- Método que dada una URL realiza una petición HTTP Get para obtener la respuesta.

```
private String inputStreamToString(final InputStream content)
          throws Exception {
     ByteArrayOutputStream bout = new ByteArrayOutputStream();
     byte[] buf = new byte[BUFFER_SIZE];
     int len = 0;
     while ((len = content.read(buf)) > 0) {
          bout.write(buf, 0, len);
     }
     content.close();
     return bout.toString();
}
```

### 5.2.5 JSON

La funcionalidad de leer ficheros en formato .json es provista por defecto por el API de Android a través del paquete “org.json”. Su uso se limita a la clase ConfigurationReader.

#### 5.2.5.1 Configuration Reader

- Está clase es la encargada de leer el fichero de configuración “*settings.json*” que define el listado de dispositivos de la casa domótica y las acciones que pueden realizar dichos dispositivos. El formato del fichero es el siguiente:

```
{
"settings": {
      "url": "settings",
      "house": {
            "id": "1",
            "name": "ciudad tecnologica",
            "zones": [
            {
                  "id": "1",
                  "name": "cocina",
                  "url": "cocina",
                  "devices": [
                  {
                        "id": "1",
                        "name": "Luz",
                        "url": "LUZ1",
                        "actions": [
                        {
                              "id": "1",
                              "name": "Encender",
                              "url": "ON"
                        },
                        {
                              "id": "2",
                              "name": "Apagar",
                              "url": "OFF"
                        }
                        ]
                  },
                  {
                        "id": "2",
                        "name": "Ventana",
                        "url": "VENTANA1",
                        "actions": [
                        {
                              "id": "3",
                              "name": "Abrir",
                              "url": "OPEN"
                        },
```

```
{
		"id": "4",
		"name": "Cerrar",
		"url": "CLOSE"
	}
	]
},
{
	"id": "3",
	"name": "Horno",
	"url": "HORNO",
	"actions": [
	{
		"id": "3",
		"name": "Encender",
		"url": "ON"
	},
	{
		"id": "4",
		"name": "Apagar",
		"url": "OFF"
	}
	]
},
{
	"id": "4",
	"name": "Puerta",
	"url": "PUERTA1",
	"actions": [
	{
		"id": "3",
		"name": "Abrir",
		"url": "OPEN"
	},
	{
		"id": "4",
		"name": "Cerrar",
		"url": "CLOSE"
	}
	]
},
```

```
                {
                      "id": "5",
                      "name": "Persiana",
                      "url": "PERSIANA",
                      "actions": [
                      {
                            "id": "5",
                            "name": "Subir",
                            "url": "UP"
                      },
                      {
                            "id": "6",
                            "name": "Bajar",
                            "url": "DOWN"
                      },
                      {
                            "id": "7",
                            "name": "Parar",
                            "url": "STOP"
                      }
                      ]
                }
                ]
          }
          ]
     }
  }
}
```

- Método encargado de leer el contenido del fichero .json del directorio *assets* de la aplicación.

```
    private String loadJSONFromAsset(final Context context) {

        String json = null;

        AssetManager mngr = context.getAssets();
        InputStream is = mngr.open(PATH);
        int size = is.available();

        byte[] buffer = new byte[size];

        is.read(buffer);
        is.close();

        json = new String(buffer, "UTF-8");
```

- Método encargado de leer el fichero .json y cargar sus datos en la estructura de objectos del sistema. En este caso lectura del tag *house*.

```
public House loadConfiguration(final Context context) {

    House house = null;
    String jsonData = loadJSONFromAsset(context);
    JSONObject json = new JSONObject(jsonData);
    JSONObject settings = json.getJSONObject(TAG_SETTINGS);

    JSONObject houseJSON = settings.getJSONObject(TAG_HOUSE);
    int idHouse = houseJSON.getInt(TAG_ID);
    String nameHouse = houseJSON.getString(TAG_NAME);
    String urlHouse = houseJSON.getString(TAG_URL);
```

- Lectura del tag *devices*.

```
JSONArray devicesJSON = zoneJSON.getJSONArray(TAG_DEVICES);
List<Device> devices = new ArrayList<Device>();

// looping through all devices
for (int j = 0; j < devicesJSON.length(); j++) {

     JSONObject deviceJSON = devicesJSON.getJSONObject(j);
     int idDevice = deviceJSON.getInt(TAG_ID);
     String nameDevice = deviceJSON.getString(TAG_NAME);
     String urlDevice = deviceJSON.getString(TAG_URL);
```

- Lectura del tag *actions*.

```
JSONArray actionsJSON = deviceJSON.getJSONArray(TAG_ACTIONS);
List<Action> actions = new ArrayList<Action>();

// looping through all actions
for (int k = 0; k < actionsJSON.length(); k++) {

     JSONObject actionJSON = actionsJSON.getJSONObject(k);
     int idAction = actionJSON.getInt(TAG_ID);
     String nameAction = actionJSON.getString(TAG_NAME);
     String urlAction = actionJSON.getString(TAG_URL);
```

- Creación de los objetos.

```
          Action action = new Action(idAction, nameAction,
                                     urlAction);
          actions.add(action);
          }
     Device device = new Device(idDevice, nameDevice, urlDevice,
                                actions);
     devices.add(device);
     }

Zone zone = new Zone(idZone, nameZone, urlZone, devices);
zones.add(zone);
}

house = new House(idHouse, nameHouse, urlHouse, zones);
```

# 6. PRUEBAS

En este capítulo se definen las pruebas de la aplicación, dividiéndolas en tres tipos: de funcionalidad, de estrés y de recuperación de errores. Debido a la falta de disponibilidad del servidor de domótica, en el entorno de pruebas se ha sustituido por un simulador que responda a las peticiones HTTP de la aplicación móvil.

## 6.1 PRUEBAS DE FUNCIONALIDAD

Estas pruebas se desarrollan con el objetivo de verificar la funcionalidad de los distintos elementos que componen el sistema. Se realizan en un entorno controlado y comprobando que la respuesta de las operaciones coincidía con los requisitos establecidos en la fase de análisis.

**Tabla 6-1: Prueba de Funcionalidad – PF_01**

| | |
|---|---|
| **Identificador** | **PF-01** |
| **Descripción** | Inicio del sistema |
| **Estado Inicial** | Aplicación instalada |
| **Estado Final** | Aplicación encendida y mostrando la imagen de la cámara |
| **Procedimiento** | Ir al menu de aplicaciones del móvil y pulsar sobre el icono de la aplicación |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_01 |

**Tabla 6-2: Prueba de Funcionalidad – PF_02**

| | |
|---|---|
| **Identificador** | **PF-02** |
| **Descripción** | Reconocimiento de un marcador |
| **Estado Inicial** | Aplicación encendida |
| **Estado Final** | La aplicación al reconocer un marcador:<br>• dibuja un círculo rojo sobre él<br>• muestra el icono de reconocimiento<br>• muestra el texto con la cuenta atrás para el procesamiento |
| **Procedimiento** | Buscar un marcador con la cámara del móvil |

| | |
|---|---|
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_02<br>REQ_SW_03<br>REQ_SW_04<br>REQ_SW_05<br>REQ_SW_08<br>REQ_SW_23<br>REQ_SW_24<br>REQ_SW_27 |

**Tabla 6-3: Prueba de Funcionalidad – PF_03**

| | |
|---|---|
| **Identificador** | **PF-03** |
| **Descripción** | Tiempo mínimo de reconocimiento |
| **Estado Inicial** | Marcador candidato reconocido |
| **Estado Final** | Marcador seleccionado listo para iniciar su procesamiento |
| **Procedimiento** | Fijar la cámara sobre el marcador el tiempo que marque la cuenta atrás. Una vez que la cuenta atrás haya llegado a 0 la aplicación:<br>• dibuja un círculo verde sobre el marcador.<br>• muestra el texto con el nombre del dispositivo<br>• pasa al modo de selección de la acción configurado |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_06<br>REQ_SW_07<br>REQ_SW_08<br>REQ_SW_10<br>REQ_SW_25<br>REQ_SW_26 |

**Tabla 6-4: Prueba de Funcionalidad – PF_04**

| | |
|---|---|
| **Identificador** | **PF-04** |
| **Descripción** | Listado de la acción modo “Texto” |
| **Estado Inicial** | Marcador previamente seleccionado<br>Tipo de interfaz seleccionada: Texto/* |
| **Estado Final** | Lista de posibles acciones a ejecutar sobre el dispositivo mediante un menú de opciones |
| **Procedimiento** | Automático, la aplicación mostrará un menu desplegable con el listado de acciones a elegir |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_10<br>REQ_SW_11<br>REQ_SW_12 |

**Tabla 6-5: Prueba de Funcionalidad – PF_05**

| | |
|---|---|
| **Identificador** | **PF-05** |
| **Descripción** | Listado de la acción modo “Voz” |
| **Estado Inicial** | Marcador previamente seleccionado<br>Tipo de interfaz seleccionada: TTS/* |
| **Estado Final** | Lista de posibles acciones a ejecutar sobre el dispositivo mediante un mensaje de voz |
| **Procedimiento** | Automático, la aplicación emitirá un mensaje por voz con el listado de acciones a elegir<br>Se mostrará el icono de TTS en la aplicación |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_10<br>REQ_SW_11<br>REQ_SW_13<br>REQ_SW_28 |

**Tabla 6-6: Prueba de Funcionalidad – PF_06**

| | |
|---|---|
| **Identificador** | **PF-06** |
| **Descripción** | Selección de la acción modo “Texto” |
| **Estado Inicial** | Marcador previamente seleccionado<br>Tipo de interfaz seleccionada: Texto/*<br>Menu con la lista de acciones desplegado |
| **Estado Final** | Acción a ejecutar sobre el dispositivo domótico seleccionado |
| **Procedimiento** | El usuario pulsará sobre el botón con el nombre de la acción que quiera ejecutar |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_10<br>REQ_SW_14<br>REQ_SW_15 |

**Tabla 6-7: Prueba de Funcionalidad – PF_07**

| | |
|---|---|
| **Identificador** | **PF-07** |
| **Descripción** | Selección de la acción mediante reconocimiento de voz |
| **Estado Inicial** | Marcador previamente seleccionado<br>Tipo de interfaz seleccionada: Recog/*<br>Menu con la lista de acciones desplegado / Listado de acciones emitido por voz |
| **Estado Final** | Acción a ejecutar sobre el dispositivo domótico seleccionado |
| **Procedimiento** | El usuario dictará a la aplicación la acción que quiere ejecutar y la aplicación obtendrá la acción mediante reconocimiento de voz.<br>Dicha acción se mostrará al usuario por el menu de acciones o mediante un comando de voz.<br>Se mostrará el icono de reconocimiento de voz. |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_10<br>REQ_SW_14 |

| | |
|---|---|
| | REQ_SW_16<br>REQ_SW_29 |

**Tabla 6-8: Prueba de Funcionalidad – PF_08**

| | |
|---|---|
| **Identificador** | **PF-08** |
| **Descripción** | Ejecución de la acción sobre un dispositivo domótico. |
| **Estado Inicial** | Marcador previamente seleccionado<br>Acción a ejecutar sobre el dispositivo seleccionada |
| **Estado Final** | Acción ejecutada sobre el dispositivo domótico |
| **Procedimiento** | Automático, una vez el usuario ha seleccionado la acción, la aplicación enviará una conexión HTTP al servidor de domótica, con el dispositivo y la acción a ejecutar.<br>Se mostrará el icono de petición al servidor de domótica |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_10<br>REQ_SW_17<br>REQ_SW_18<br>REQ_SW_19<br>REQ_SW_30<br>REQ_SW_35 |

**Tabla 6-9: Prueba de Funcionalidad – PF_09**

| | |
|---|---|
| **Identificador** | **PF-09** |
| **Descripción** | Resultado de la acción en modo Texto |
| **Estado Inicial** | Marcador previamente seleccionado<br>Acción ejecutada sobre el dispositivo domótico |
| **Estado Final** | Mensaje de texto con el resultado de la acción |

| | |
|---|---|
| **Procedimiento** | Automático, una vez la aplicación haya obtenido el resultado de la ejecución de la acción, mostrará un mensaje con el resultado. |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_10<br>REQ_SW_20<br>REQ_SW_21 |

**Tabla 6-10: Prueba de Funcionalidad – PF_10**

| | |
|---|---|
| **Identificador** | **PF-10** |
| **Descripción** | Resultado de la acción por voz |
| **Estado Inicial** | Marcador previamente seleccionado<br>Acción ejecutada sobre el dispositivo domótico |
| **Estado Final** | Mensaje de voz con el resultado de la acción |
| **Procedimiento** | Automático, una vez la aplicación haya obtenido el resultado de la ejecución de la acción, se emitirá un mensaje de voz con el resultado de la acción. |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_10<br>REQ_SW_20<br>REQ_SW_22 |

**Tabla 6-11: Prueba de Funcionalidad – PF_11**

| | |
|---|---|
| **Identificador** | **PF-11** |
| **Descripción** | Configuración del tipo de interfaz |
| **Estado Inicial** | N.A. |
| **Estado Final** | Modo de interfaz modificado |
| **Procedimiento** | El usuario podrá modificar el tipo de interfaz a utilizar mediante el menú de opciones de la aplicación.<br>Podrá elegir entre interfaz de texto, TTS y |

| | |
|---|---|
| | reconocimiento de voz. |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_31<br>REQ_SW_32 |

**Tabla 6-12: Prueba de Funcionalidad – PF_12**

| | |
|---|---|
| **Identificador** | **PF-12** |
| **Descripción** | Configuración del tiempo del marcador |
| **Estado Inicial** | N.A. |
| **Estado Final** | Tiempo de la cuenta atrás para seleccionar el marcador modificado. |
| **Procedimiento** | El usuario podrá modificar el tiempo necesario para elegir el marcador mediante el menú de opciones de la aplicación.<br>Podrá elegir entre un rango de 1 a 5 segundos. |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_31<br>REQ_SW_33 |

**Tabla 6-13: Prueba de Funcionalidad – PF_13**

| | |
|---|---|
| **Identificador** | **PF-13** |
| **Descripción** | Configuración del servidor de domótica |
| **Estado Inicial** | N.A. |
| **Estado Final** | Dirección IP y puerto del servidor de domótica modificados. |
| **Procedimiento** | El usuario podrá modificar la dirección IP y el puerto de conexión al servidor de domótica. |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_31<br>REQ_SW_34 |

## 6.2 PRUEBAS DE ESTRÉS

Estas pruebas tienen como objetivo comprobar el funcionamiento del sistema bajo una gran carga de peticiones.

**Tabla 6-14: Prueba de Estrés – PS_01**

| | |
|---|---|
| **Identificador** | **PS-01** |
| **Descripción** | Procesamiento continuo con mínimo tiempo de reconocimiento. |
| **Estado Inicial** | Cuenta atrás para reconocer el marcador establecida a 1 segundo (ver PF-12).<br>Modo de la interfaz: Texto/TTS/Recog (ver PF-11) |
| **Estado Final** | Marcador procesado de manera correcta en todos los casos. |
| **Procedimiento** | Reconocer un marcador y ejecutar la acción a través de la interfaz de texto.<br>Repetir el proceso de forma continua 10 veces sin separar la cámara del móvil del marcador. |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_03<br>REQ_SW_07<br>REQ_SW_08<br>REQ_SW_10<br>REQ_SW_12<br>REQ_SW_15<br>REQ_SW_17<br>REQ_SW_18<br>REQ_SW_19<br>REQ_SW_20<br>REQ_SW_33 |

**Tabla 6-15: Prueba de Estrés – PS_02**

| | |
|---|---|
| **Identificador** | **PS-02** |
| **Descripción** | Marcadores múltiples. |

| | |
|---|---|
| **Estado Inicial** | Varios marcadores capturados en la aplicación |
| **Estado Final** | Marcador elegido es el marcador más cercano al centro de la imagen de la cámara. |
| **Procedimiento** | Colocar dos marcadores cercanos entre sí para que la aplicación reconozca los dos a la vez. |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_09 |

## 6.3 PRUEBAS DE RECUPERACIÓN DE ERRORES

Las pruebas descritas a continuación verifican el comportamiento de la aplicación en caso de errores.

**Tabla 6-16: Prueba de Recuperación de Error – PE_01**

| | |
|---|---|
| **Identificador** | **PE-01** |
| **Descripción** | Acción no seleccionada |
| **Estado Inicial** | Marcador seleccionado<br>Tipo de interfaz: Texto/* |
| **Estado Final** | Mensaje de error “Comando no reconocido” |
| **Procedimiento** | Una vez desplegado el menú con las acciones a ejecutar, no pulsar ninguna acción en el plazo de 5 segundos. La aplicación mostrará un mensaje de error. |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_10<br>REQ_SW_11<br>REQ_SW_12<br>REQ_SW_14<br>REQ_SW_15<br>REQ_SW_20<br>REQ_SW_21<br>REQ_SW_22 |

**Tabla 6-17: Prueba de Recuperación de Error – PE_02**

| | |
|---|---|
| **Identificador** | **PE-02** |
| **Descripción** | Acción no dictada |
| **Estado Inicial** | Marcador seleccionado<br>Tipo de interfaz: Recog/* |
| **Estado Final** | Mensaje de error “Comando no reconocido” |
| **Procedimiento** | Una vez iniciado el proceso de reconocimiento de voz, no dictar ninguna acción en el plazo de 5 segundos. La aplicación mostrará/dictará un mensaje de error. |
| **Resultado** | Éxito |
| **Requisito Software** | REQ_SW_10<br>REQ_SW_11<br>REQ_SW_12<br>REQ_SW_14<br>REQ_SW_16<br>REQ_SW_20<br>REQ_SW_21<br>REQ_SW_22 |

**Tabla 6-18: Prueba de Recuperación de Error – PE_03**

| | |
|---|---|
| **Identificador** | **PE_03** |
| **Descripción** | Error de conexión con el servidor de domótica |
| **Estado Inicial** | Marcador previamente seleccionado<br>Acción a ejecutar sobre el dispositivo seleccionada |
| **Estado Final** | Mensaje de error “Comando incorrecto, fallo en el servidor” |
| **Procedimiento** | Deshabilitar el servidor de domótica utilizado en las pruebas para que no haya conexión desde la aplicación. La aplicación mostrará un mensaje de error. |
| **Resultado** | Éxito |

| | |
|---|---|
| **Requisito Software** | REQ_SW_17 |
| | REQ_SW_18 |
| | REQ_SW_19 |
| | REQ_SW_20 |

## 6.4 MATRIZ DE TRAZABILIDAD DE PRUEBAS

A continuación se describe la matriz de trazabilidad de los requisitos software contra las pruebas. Esta matriz verifica que el plan de pruebas del sistema cubre todos los requisitos software del proyecto.

**Tabla 6-19: Matriz de Trazabilidad Requisitos Software - Pruebas**

| | PF_01 | PF_02 | PF_03 | PF_04 | PF_05 | PF_06 | PF_07 | PF_08 | PF_09 | PF_10 | PF_11 | PF_12 | PF_13 | PS_01 | PS_02 | PE_01 | PE_02 | PE_03 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **REQ_SW_01** | X | | | | | | | | | | | | | | | | | |
| **REQ_SW_02** | | X | | | | | | | | | | | | | | | | |
| **REQ_SW_03** | | X | | | | | | | | | | | | X | | | | |
| **REQ_SW_04** | | X | | | | | | | | | | | | | | | | |
| **REQ_SW_05** | | X | | | | | | | | | | | | | | | | |
| **REQ_SW_06** | | | X | | | | | | | | | | | | | | | |
| **REQ_SW_07** | | | X | | | | | | | | | | | X | | | | |
| **REQ_SW_08** | | X | X | | | | | | | | | | | X | | | | |
| **REQ_SW_09** | | | | | | | | | | | | | | | X | | | |
| **REQ_SW_10** | | | X | X | X | X | X | X | X | X | | | | X | | X | X | |
| **REQ_SW_11** | | | | X | X | | | | | | | | | | | X | X | |
| **REQ_SW_12** | | | | X | | | | | | | | | | X | | X | X | |
| **REQ_SW_13** | | | | | X | | | | | | | | | | | | | |
| **REQ_SW_14** | | | | | | X | X | | | | | | | | | X | X | |
| **REQ_SW_15** | | | | | | X | | | | | | | | X | | X | | |
| **REQ_SW_16** | | | | | | | X | | | | | | | | | | X | |
| **REQ_SW_17** | | | | | | | | X | | | | | | X | | | | X |
| **REQ_SW_18** | | | | | | | | X | | | | | | X | | | | X |
| **REQ_SW_19** | | | | | | | | X | | | | | | X | | | | X |
| **REQ_SW_20** | | | | | | | | | X | X | | | | X | | X | X | X |
| **REQ_SW_21** | | | | | | | | | X | | | | | | | X | X | |
| **REQ_SW_22** | | | | | | | | | | X | | | | | | X | X | |
| **REQ_SW_23** | | X | | | | | | | | | | | | | | | | |
| **REQ_SW_24** | | X | | | | | | | | | | | | | | | | |
| **REQ_SW_25** | | | X | | | | | | | | | | | | | | | |
| **REQ_SW_26** | | | X | | | | | | | | | | | | | | | |
| **REQ_SW_27** | | X | | | | | | | | | | | | | | | | |
| **REQ_SW_28** | | | | | X | | | | | | | | | | | | | |
| **REQ_SW_29** | | | | | | | X | | | | | | | | | | | |
| **REQ_SW_30** | | | | | | | | X | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **REQ_SW_31** | | | | | | | | | | | X | X | X | | | | | |
| **REQ_SW_32** | | | | | | | | | | | X | | | | | | | |
| **REQ_SW_33** | | | | | | | | | | | | X | | X | | | | |
| **REQ_SW_34** | | | | | | | | | | | | | X | | | | | |
| **REQ_SW_35** | | | | | | | | X | | | | | | | | | | |

# 7. GESTIÓN DEL PROYECTO

Este capítulo describe todos los aspectos relacionados con la gestión llevada a cabo sobre el presente proyecto fin de carrera tales como metodología, ciclo de vida, recursos y costes.

## 7.1 METODOLOGÍA

La metodología de Ingeniería de Software hace referencia a como han de obtenerse los distintos productos parciales y finales durante el ciclo de vida de una aplicación, es decir, proponen un proceso disciplinado en cuanto a la ejecución de la misma se refiere, realizando especial énfasis en su planificación y control.

En el caso que nos atañe, se tomo la decisión de utilizar una metodología orientada a objetos basada en UML 2.0 Se tomo la decisión de utilizar la especificación UML ya que era la más adecuada a las necesidades del proyecto, ya que dota del lenguaje de modelado necesario para especificar o describir los métodos y/o procesos que debía llevar acabo la aplicación.

## 7.2 CICLO DE VIDA

El desarrollo del proyecto ha seguido un modelo de ciclo de vida en cascada clásico. Este consiste en dividir el desarrollo de la aplicación en una serie de etapas que se llevarán acabo de manera consecutiva, estas se describen más detalladamente a continuación.

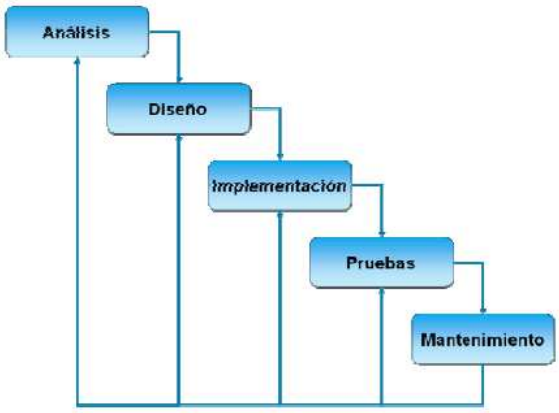


**Figura 7-1: Ciclo de Vida en Cascada**

- Obtención de requisitos (Análisis)
    - Proceso de obtención de las necesidades del cliente, las especificaciones que deberá satisfacer la aplicación.

- Diseño de la aplicación
    - Consistente en el proceso de abstracción de todas las necesidades obtenidas del proceso de obtención de requisitos.
- Implementación de la aplicación
    - Proceso de implementación del código fuente de la aplicación en un determinado lenguaje de programación.
- Pruebas de la aplicación
    - Etapa en la que el equipo de desarrollo se encargará de testear el correcto funcionamiento de cada uno de los módulos de la aplicación.
- Mantenimiento de la aplicación
    - Es una de las etapas más críticas, ya que en ella, el usuario entra en contacto directo con la aplicación, y pueden detectarse errores en el funcionamiento de la misma, así como nuevos requisitos a desarrollar.

El ciclo de vida en Cascada ha seguido el modelo clásico, en el que las etapas que se mencionaron previamente se siguen de manera secuencial.

## 7.3 PLANIFICACIÓN DEL PROYECTO

### 7.3.1 Listado de Actividades

En esta sección se trata el proceso de planificación seguido para la realización del proyecto.

A continuación se muestra el listado de actividades o fases del proyecto y el tiempo asociado a la realización de cada una de ellas:

- Fecha de inicio de la actividad.
- Fecha de fin de la actividad:
- Número de horas dedicadas a esa actividad

**Tabla 7-1: Actividades del Proyecto**

| Fase | Inicio | Fin | Horas |
|---|---|---|---|
| **Análisis** | 17/12/2012 | 28/02/2013 | 164 |
| **Diseño** | 01/03/2013 | 31/03/2013 | 60 |
| **Implementación** | 01/04/2013 | 26/05/2013 | 172 |
| **Pruebas** | 27/05/2013 | 31/05/2013 | 20 |
| **Documentación** | 01/06/2013 | 24/06/2013 | 64 |
| **Total** | 17/12/2012 | 24/06/2013 | **480** |

### 7.3.2 Diagrama de Gantt

La planificación de actividades vista en el apartado anterior, se puede representar mediante un diagrama de Gantt como el que se muestra a continuación.

| | Nombre | Inicio | Terminado |
|---|---|---|---|
| 1 | Análisis | 17/12/12 | 28/02/13 |
| 2 | Diseño | 1/03/13 | 29/03/13 |
| 3 | Implementación | 1/04/13 | 24/05/13 |
| 4 | Pruebas | 27/05/13 | 31/05/13 |
| 5 | Documentación | 3/06/13 | 24/06/13 |

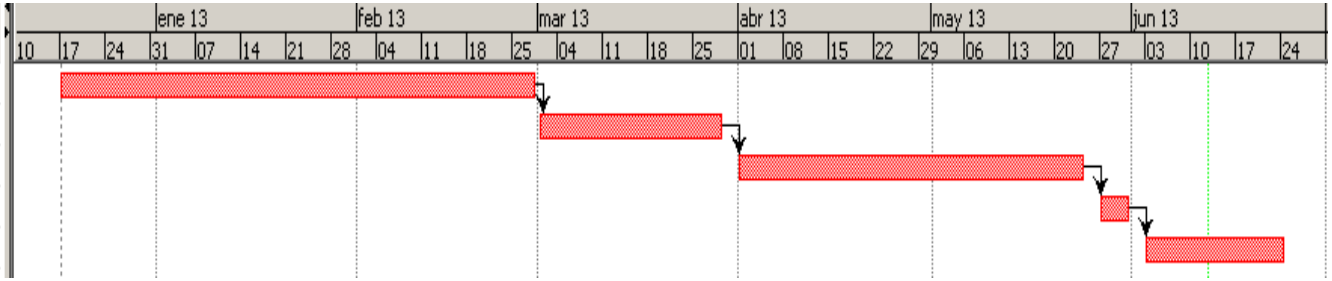


**Figura 7-2: Diagrama de Gantt**

## 7.4 ORGANIGRAMA

Este apartado se encargará de detallar todos los recursos humanos para llevar acabo este proyecto.

En la siguiente figura se puede observar el organigrama utilizado en este proyecto: un jefe de proyecto encargado de la gestión del proyecto, la toma de decisiones y la documentación final; un analista, encargado de la toma de requisitos, y el diseño de la aplicación; y por ultimo, un programador, encargado de la implementación de los diseños aportados por el analista.

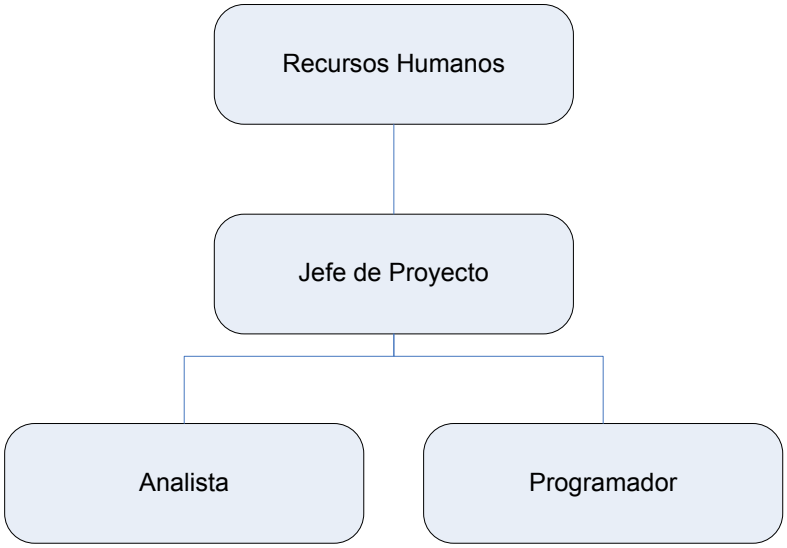


**Figura 7-3: Organigrama del Proyecto**

## 7.5 PRESUPUESTO

El presupuesto se divide en coste de personal, coste hardware y coste software.

### 7.5.1 Coste de Personal

Se ha de calcular el coste de personal dependiendo del rol desempeñado y el tiempo invertido por cada uno de ellos en el desarrollo del proyecto

**Tabla 7-2: Coste de Personal**

| Perfil | Horas Análisis | Horas Diseño | Horas Implementación | Horas Pruebas | Horas Doc | Horas Totales | Coste Hora | Coste Total |
|---|---|---|---|---|---|---|---|---|
| **Jefe de Proyecto** | 40 | 16 | 8 | 4 | 64 | 132 | 70,00 € | 9.240,00 € |
| **Analista** | 124 | 44 | 16 | 0 | 0 | 184 | 40,00 € | 7.360,00 € |
| **Programador** | 0 | 0 | 148 | 16 | 0 | 164 | 25,00 € | 4.100,00 € |
| | | | | | | | **Total** | **20.700,00 €** |

### 7.5.2 Coste Hardware

Para el desarrollo del proyecto, ha sido necesaria la adquisición de un portátil y un dispositivo móvil que utilizarán en cada una de las fases del proyecto.

A continuación se detallan los datos de ambos dispositivos, para el cual se ha calculado el coste imputable a partir de la fórmula de amortización descrita abajo.

**Tabla 7-3: Coste Hardware**

| Descripción | Coste | % Uso dedicado | Dedicación (meses) | Periodo de depreciación (meses) | Coste imputable |
|---|---|---|---|---|---|
| **Portatil Asus Eee PC T101MT** | 364,00 € | 100,00% | 6 | 60 | 36,40 € |
| **Dispositivo móvil Nexus 4** | 255,00 € | 100,00% | 6 | 60 | 25,50 € |
| | | | | **Total** | **61,90 €** |

Fórmula de cálculo de la amortización:
(A / B) x C x D

A = nº de meses desde la fecha de facturación en que el equipo es utilizado.
B = periodo de depreciación (60 meses).
C = coste del equipo (sin IVA).

D = % del uso que se dedica al proyecto (habitualmente 100%).

### 7.5.3 Coste Software

A pesar de que la mayor parte del software con el que se ha trabajado es de código libre, y por tanto no tienen coste alguno, ha sido necesaria la adquisición de una licencia software.

**Tabla 7-4: Coste Software**

| Descripción | Coste (€) |
|---|---|
| **Microsoft Office Professional 2010** | 499,00 € |
| **Total** | **499,00 €** |

### 7.5.4 Resumen de Costes

Tras analizar los distintos costes asociados al proyecto, se resume a continuación cada uno de ellos y se establece el coste total del proyecto. Se detalla el coste total con y sin I.V.A.

**Tabla 7-5: Resumen de Costes**

| Descripción | Coste (Euros) |
|---|---|
| **Coste de Personal** | 20.700,00 € |
| **Coste Hardware** | 61,90 € |
| **Coste Software** | 499,00 € |
| **SubTotal** | 21.260,90 € |
| **I.V.A (21%)** | 4.464,79 € |
| **Total** | **25.725,69 €** |

# 8. CONCLUSIONES Y LÍNEAS FUTURAS

Este capítulo contiene las conclusiones obtenidas del desarrollo de este proyecto, tomando como referencia el objetivo inicial que tenia el mismo, y el resultado final obtenido, a través de su proceso de desarrollo, así como los posibles trabajos futuros que se podrían llevar a cabo sobre el mismo.

## 8.1 CONCLUSIONES DEL PROYECTO

El objetivo principal del proyecto, la generación de un sistema móvil con el que una persona de movilidad reducida pueda utilizar un electrodoméstico de una casa domótica al reconocer una imagen, ha sido completado de forma satisfactoria.

Todos los objetivos básicos para el proyecto se han cubierto en su totalidad:

- Desarrollo de una aplicación móvil mediante la plataforma Android

  Se ha implementado una aplicación a través del SDK de Android 4.2.2 (API 17).

- Reconocimiento de imágenes o etiquetas mediante la cámara del móvil

  La aplicación reconoce marcadores en blanco y negro de 7’5x7’5 cm a una distancia máxima de 3’5 y 4 metros.

- Utilización de un sistema de realidad aumenta para mostrar información “aumentada” al usuario

  Aunque no se ha utilizado un framework existente de realidad aumentada (ver capítulo 2.3.4), la propia aplicación muestra información extra al usuario sobre la imagen recibida por la cámara, es decir se aumenta la realidad del sistema.

- Interactuar con el sistema de domótica existente para la ejecución de controles sobre los dispositivos.

  La aplicación permite ejecutar distintas acciones sobre los dispositivos domóticos existentes mediante la interfaz que ofrece el servidor de domótica.

- Proveer interfaces de voz al usuario, tanto síntesis de voz (*Text To Speech*) como reconocimiento de voz.

  La aplicación ofrece para la selección de una acción la posibilidad de reconocer voz, También se permite al usuario el escuchar mensajes de voz con las acciones a ejecutar.

Otras funcionalidades se han añadido como la utilización de una interfaz de tipo textual así como la posibilidad de poder configurar el tipo de interfaces, ya sea de voz o de texto, que suponen un valor añadido respecto a los objetivos marcados al comienzo del proyecto.

## 8.2 FUTURAS LÍNEAS DE TRABAJO

Este apartado expone algunos de los trabajos que se podrían llevar a cabo en un futuro, siguiendo la línea de este proyecto fin de carrera:

- Reducir el tamaño de los marcadores: el marcador utilizado tiene un tamaño de 7'5x7'5 centímetros que permite una detección hasta unos 3'5 y 4 metros. Una posible mejora sería intentar reducir el tamaño del marcador sin que se viese afectada la distancia de detección.
- Gestor para la creación de los marcadores en la propia aplicación: la aplicación podría incluir un módulo para la creación de marcadores que imprima los marcadores seleccionados en un PDF o en JPG para la posterior utilización del usuario.
- Utilización de un modelado 3D: para aumentar la realidad del sistema se ha utilizado un modelado en 2D relativamente simple. Una mejora en el sistema es el uso de modelados 3D más complejos, que aporten por ejemplo información o estado del dispositivo, etc.
- Algoritmo de detección más robusto ante:
    - Cambios de movimiento: el algoritmo de detección permite el *tracking* del marcador moviendo la cámara a una velocidad limitada. Una mejora sería robustecer el algoritmo para que detecte marcadores ante movimientos más bruscos y veloces.
    - Cambios de luz: otra mejora sería adaptar los *thresholds* a utilizar según el tipo de luz que se reciba, ya que ahora mismo el *threshold* utilizado es fijo y no dinámico.
- Posibilidad de hacer zoom en la cámara: actualmente la versión de OpenCV utilizada (2.4.5) no permite hacer zoom sobre las imágenes de la cámara. Una línea futura de investigación bastante interesante sería el desarrollo de esta funcionalidad.
- Pruebas reales contra el servidor domótico: debido a la no disponibilidad del servidor de domótica, las pruebas se han realizado contra un simulador. Un punto importante en el futuro sería la realización de un conjunto de pruebas completo contra el servidor de domótica real.

# 9. GLOSARIO

Esta sección contiene el glosario del proyecto.

**Tabla 9-1: Glosario**

| | |
|---|---|
| **6DOF** | Six Degrees of Freedom |
| **AAC** | Advanced Audio Coding |
| **API** | Application Program Interface |
| **AVC** | Advanced Video Coding |
| **BSD** | Berkeley Software Distribution |
| **DVI** | DirectVoice Input |
| **EHS** | European Home System |
| **EIB** | Element Interconnect Bus |
| **GPS** | Global Positioning System |
| **GSM** | Global System for Mobile Communications |
| **HMD** | Head Mounted Display |
| **HMM** | Hidden Markov Models |
| **HTML** | HyperText Markup Language |
| **HTTP** | Hypertext Transfer Protocol |
| **IOS** | iPhone Operating System |
| **IP** | Internet Protocol |
| **JSON** | JavaScript Object Notation |
| **LVCSR** | Large vocabulary continuous speech recognition |
| **MIT** | Massachussets Institute of Technology |
| **OHA** | Open Handset Alliance |
| **PC** | Personal Computer |
| **PDA** | Personal Digital Assistant |
| **PDF** | Portable Document Format |
| **PSOLA** | Pitch-Synchronous Over-Lap and Add |
| **RA** | Realidad Aumentada |
| **SDK** | Software Development Kit |
| **SQL** | Search and Query Language |
| **TTS** | Text To Speech |
| **UML** | Unified Modeling Language |

| | |
|---|---|
| **URL** | User Resource Link |
| **WER** | Word Error Rate |
| **WIFI** | Wireless Fidelity |

# 10. BIBLIOGRAFÍA

A continuación se muestra la bibliografía del proyecto.

**Tabla 10-1: Bibliografía del Proyecto**


| | |
|---|---|
| [1] | Cuota de mercado de Sistemas Operativos para móviles entre 2007 y 2011 http://blog.seattlepi.com/microsoft/chart-mobile-os-market-shares/ |
| [2] | Página oficial de WebKit, http://www.webkit.org |
| [3] | http://es.wikipedia.org/wiki/S%C3%ADntesis_de_habla |
| [4] | An introduction to text-to-speech synthesis [Dutoit, Thierry – 1997 Kluwer Academic] |
| [5] | http://ict.udlap.mx/people/ingrid/Clases/IS412/index.html |
| [6] | Speech Recognition: Statistical Methods (L R Rabiner, Rutgers University, B-H Juang, Georgia Institute of Technology) – 2006 |
| [7] | Statistical methods for speech recognition [Jelinek, Frederick – 1998 The MIT Press] |
| [8] | http://advancedtech.wordpress.com/2008/07/03/modelos-ocultos-de-markov-arquitectura/ |
| [9] | Azuma, R. (1997). A survey of augmented reality. Presence: Teleoperators and Virtual Environments, |
| [10] | Milgram, P., Takemura, H., Utsumi, A., Kishino, F. (1994) Augmented Reality: A class of displays on the reality-virtuality continuum. Proceedings of Telemanipulator and Telepresence Technologies. |
| [11] | Lee, T., Höllerer, T. (2007) Handy AR: Markerless Inspection of Augmented Reality Objects Using Fingertip Tracking. IEEE International Symposium on Wearable Computers (ISWC '07), |
| [12] | Wagner, D., Schmalstieg, D. (2009) History and future of tracking for mobile phone augmented reality. International Symposium on Ubiquitous Virtual Reality |
| [13] | Clemens, A., Schmalstieg, D. (2011) Challenges of Large-Scale Augmented Reality on Smartphones, Workshop: Enabling Large-Scale Outdoor Mixed Reality and Augmented Reality, International Symposium on Mixed and Augmented Reality (ISMAR'11), disponible para descarga en: http://www.navteq.com/outdoor_mar2011/challenges.pdf |
| [14] | Kato, H., Billinghurst, M.(1999) Marker tracking and HMD calibration for a video-based augmented reality conferencing system, Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR 99), |
| [15] | Mixare. http://www.mixare.org/, |
| [16] | Proyecto AndAR - Android Augmented Reality. http://code.google.com/p/andar/ |
| [17] | NyARToolkit. http://nyatla.jp/nyartoolkit/wp/?page_id=198 |
| [18] | SDK de Vuforia para Android, disponible para descarga en: https://ar.qualcomm.at/qdevnet/sdk/android |
| [19] | SDK de Metaio para dispositivos móviles. http://www.metaio.com/software/mobile-sdk/ |
| [20] | Proyecto ArUco: http://code.google.com/p/aruco-android/ |
| [21] | OpenCV: http://opencv.org/ |

# 11. ANEXO A: MANUAL DE INSTALACIÓN

## 11.1 INSTALACIÓN DE LA LIBRERÍA OPENCV MANAGER

Este apartado contiene la instalación de la librería OpenCV Manager desde Google Play. Esta librería es un prerequisito para la utilización del API de OpenCV en el proyecto.

1. Buscar en Google Play la librería opencv manager.

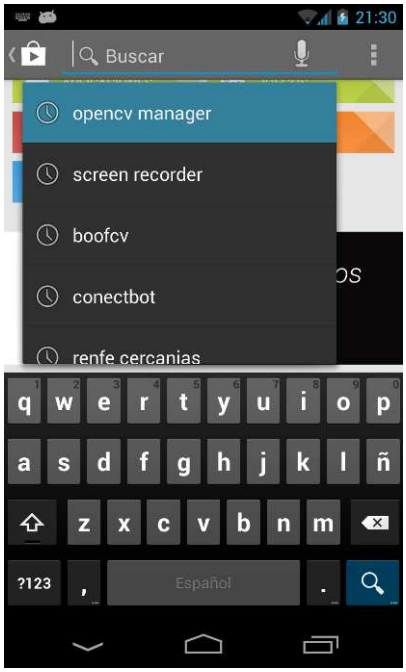


**Figura 11-1: Búsqueda opencv manager**

2. Elegir la aplicación OpenCV Manager.

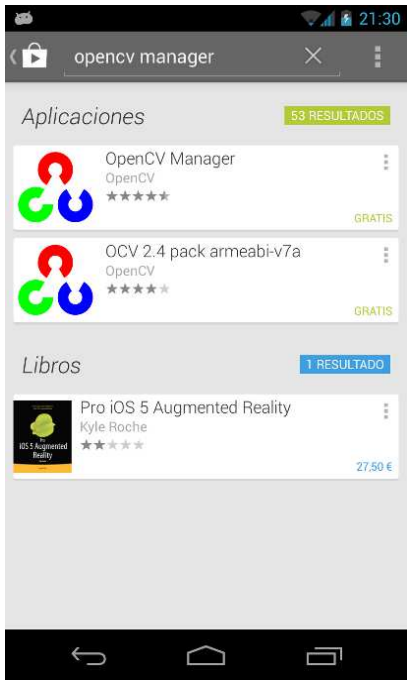


**Figura 11-2: Aplicación OpenCV Manager**

3. Instalar la aplicación OpenCV Manager.


**Figura 11-3: Instalar la aplicación OpenCV Manager**

4. Permisos de la aplicación OpenCV Manager.

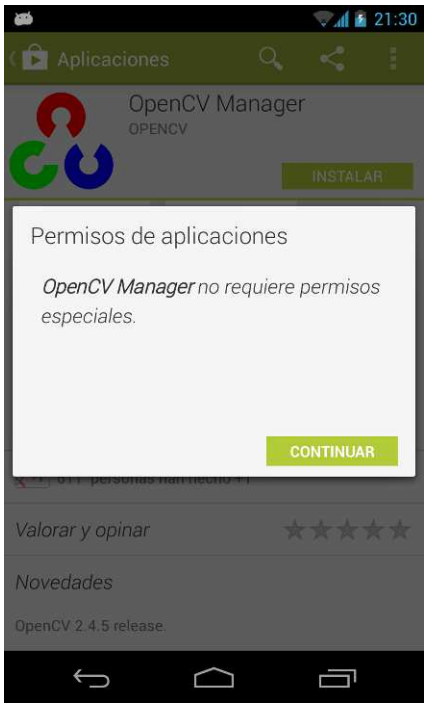


**Figura 11-4: Permisos OpenCV Manager**

5. Proceso de descarga de la aplicación OpenCV Manager.


**Figura 11-5: Proceso de descarga de OpenCV Manager**

6. Instalación finalizada de la aplicación OpenCV Manager.


**Figura 11-6: Instalación finalizada OpenCV Manager**

7. Librería OpenCV Manager.

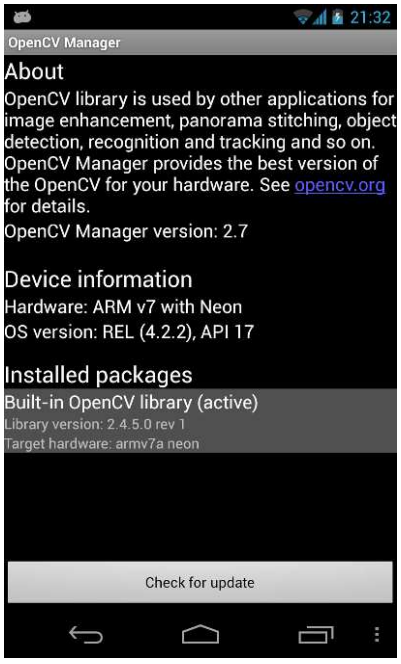


**Figura 11-7: Librería OpenCV Manager**

## 11.2INSTALACIÓN DE LA LIBRERÍA OPENCV MANAGER

Este apartado contiene la instalación de la aplicación del proyecto.

1. Copiar la aplicación del proyecto PfcUc3mAndroid.apk de la carpeta origen a una carpeta del teléfono móvil mediante un cable usb.

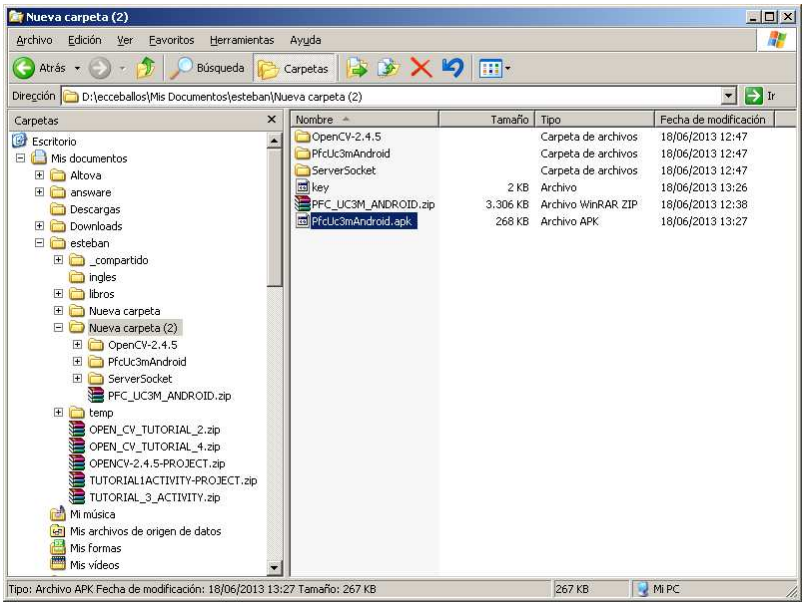


**Figura 11-8: Carpeta origen aplicación**

2. Carpeta destino en el móvil a copiar la aplicación:

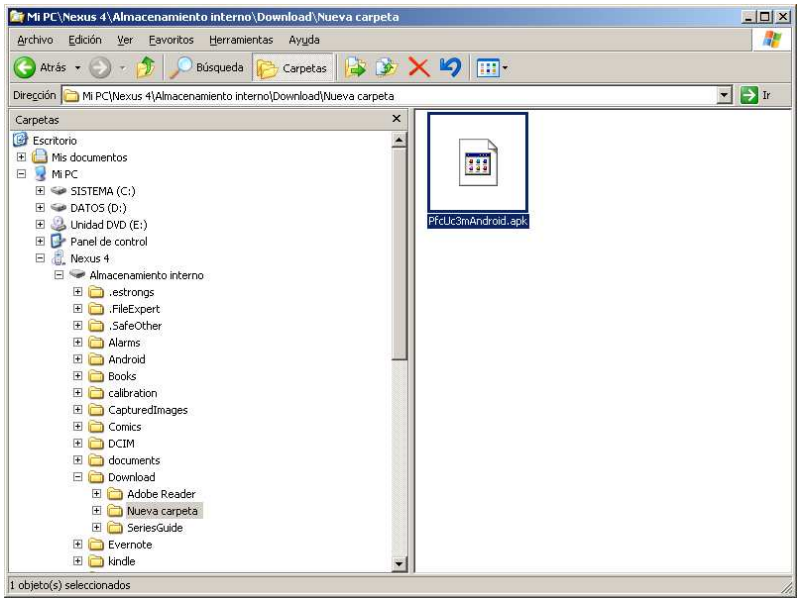


**Figura 11-9: Carpeta destino**

3. Abrir un programa de explorador de archivos en el móvil y navegar a la carpeta destino:


**Figura 11-10: Explorador de archivos móvil**

4. Abrir el fichero apk de la aplicación para su instalación:

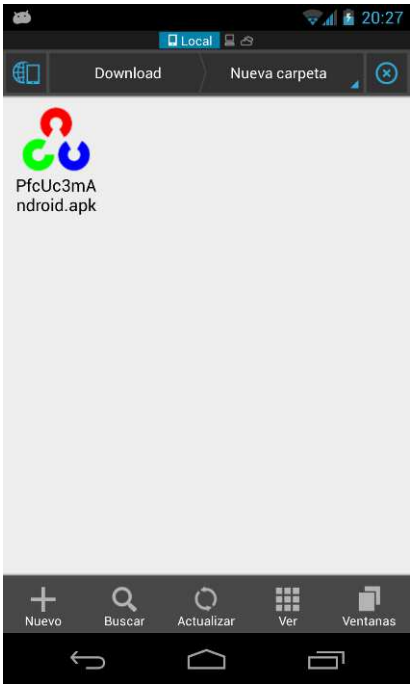


**Figura 11-11: Fichero apk para su instalación**

5. Diálogo de instalación con los permisos de la aplicación.

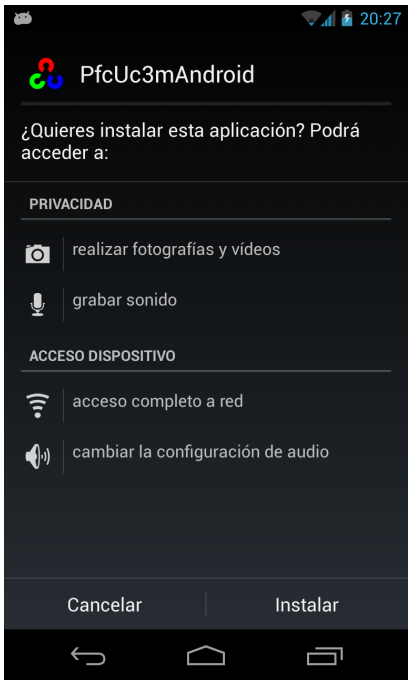


**Figura 11-12: Diálogo de instalación de la aplicación**

6. Instalación de la aplicación completada.

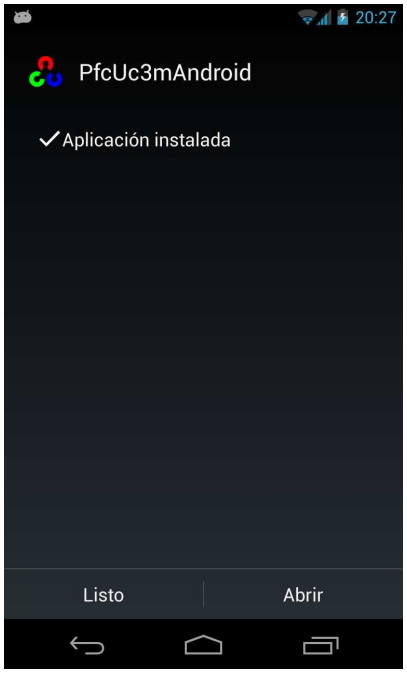


**Figura 11-13: Aplicación instalada con éxito**

# 12. ANEXO B: MANUAL DE USUARIO

A continuación se muestran unas capturas de la aplicación mostrando las funcionalidades descritas a lo largo del proyecto.

1. La aplicación muestra la imagen de la cámara en todo momento. En este caso se muestra el marcador encontrado.

**Figura 12-1: Marcador encontrado**

2. La aplicación reconoce el marcador, dibuja un círculo rojo alrededor y muestra la cuenta atrás para el procesado.


**Figura 12-2: Marcador candidato**

3. Una vez que la cuenta atrás ha finalizado, se muestra un círculo verde alrededor del marcador y se informa del dispositivo encontrado.

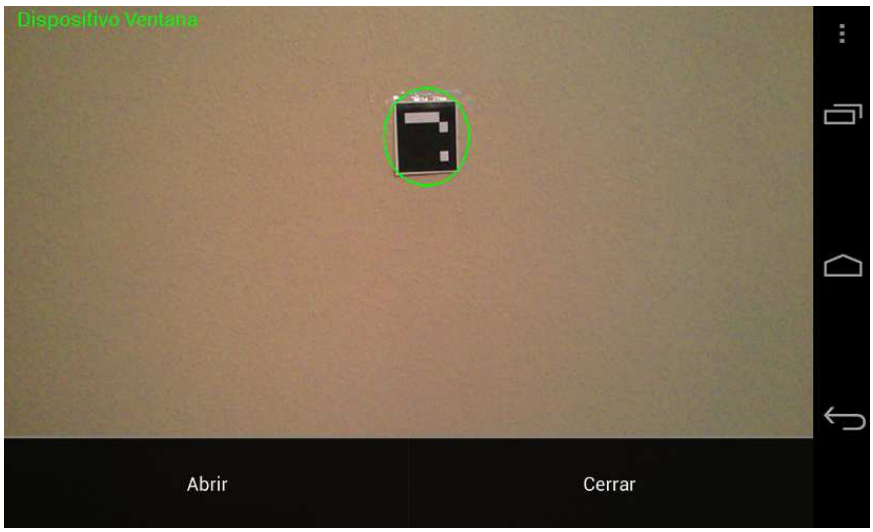


**Figura 12-3: Marcador seleccionado**

4. En este caso se muestra el listado de acciones a ejecutar mediante un menú para que el usuario pulse la pantalla del móvil.

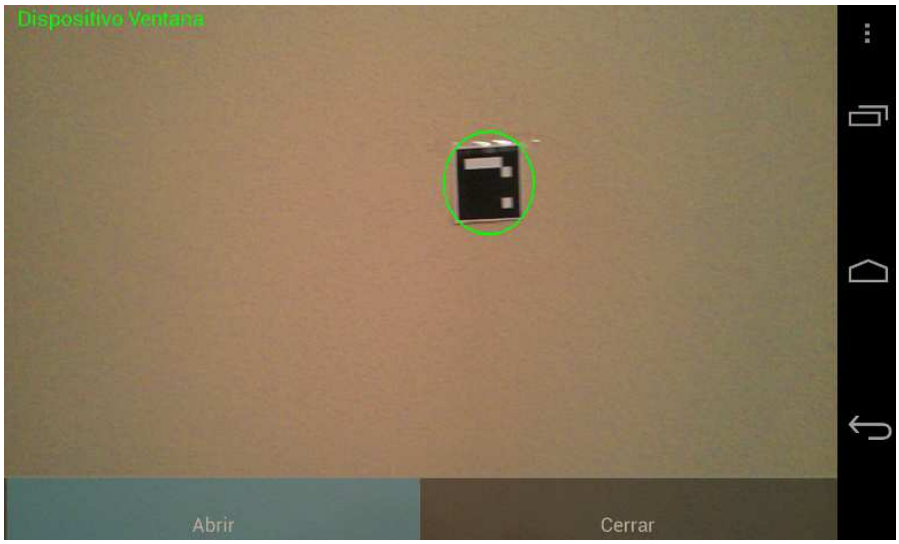


**Figura 12-4: Listado de acciones por menú**

5. Un vez elegida la acción la aplicación se conecta al servidor de domótica.

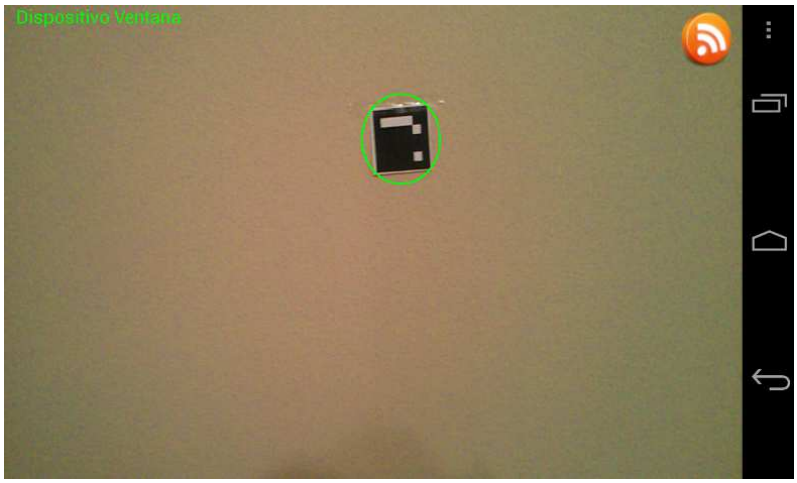


**Figura 12-5: Conexión con el servidor de domótica**

6. A continuación se muestra el resultado de la acción, en este caso mediante un pop-up.


**Figura 12-6: Mensaje de información del resultado**

7. En este ejemplo se muestra una vez seleccionado un marcador, el icono de Text To Speech (además del menú de acciones), momento en el cual se listarán las acciones medainte síntesis de voz.

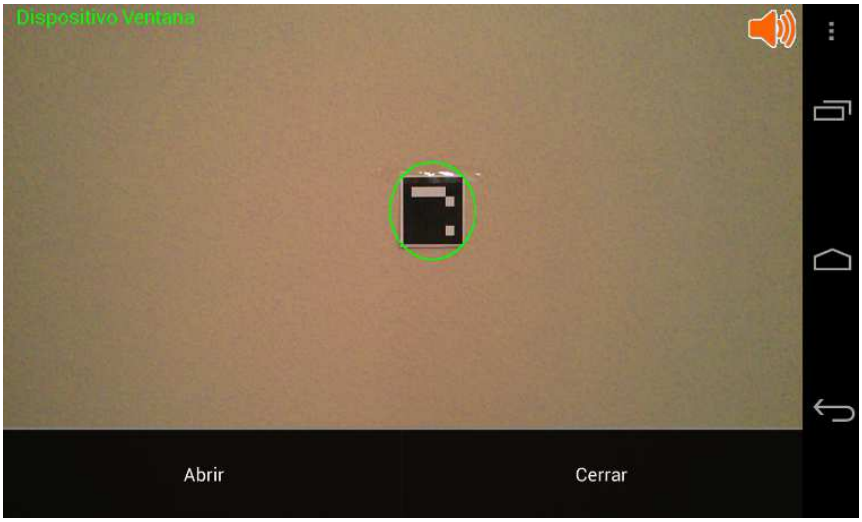


**Figura 12-7: Acciones mediante Text To Speech y texto**

8. Cuando el icono de reconocimiento de voz se muestra el usuario puede elegir la acción a ejecutar (en este caso también se puede elegir la acción por texto).

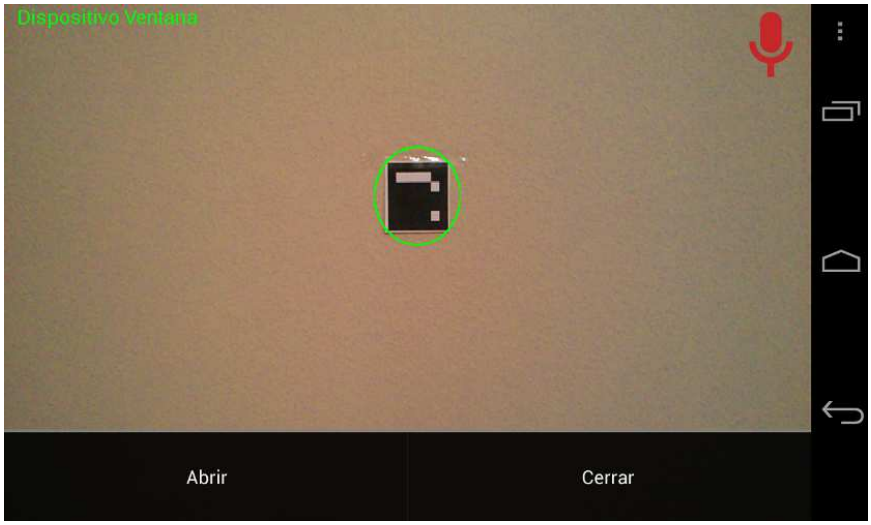


**Figura 12-8: Reconocimiento de la acción por voz y texto**

9. Información con el resultado por síntesis de voz (en este caso error al no reconocer una palabra), además de texto.


**Figura 12-9: Mensaje de información por texto y voz**

10. En este ejemplo se muestra una vez seleccionado un marcador el icono de Text To Speech, momento en el cual la aplicación dicta las acciones.

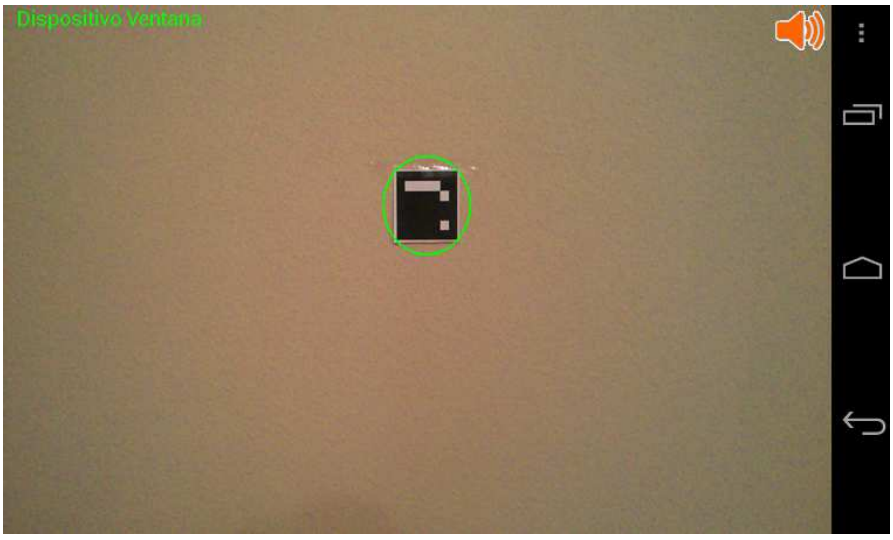


**Figura 12-10: Listado de acciones por voz**

11. Cuando el icono de reconocimiento de voz se muestra el usuario puede elegir la acción a ejecutar.

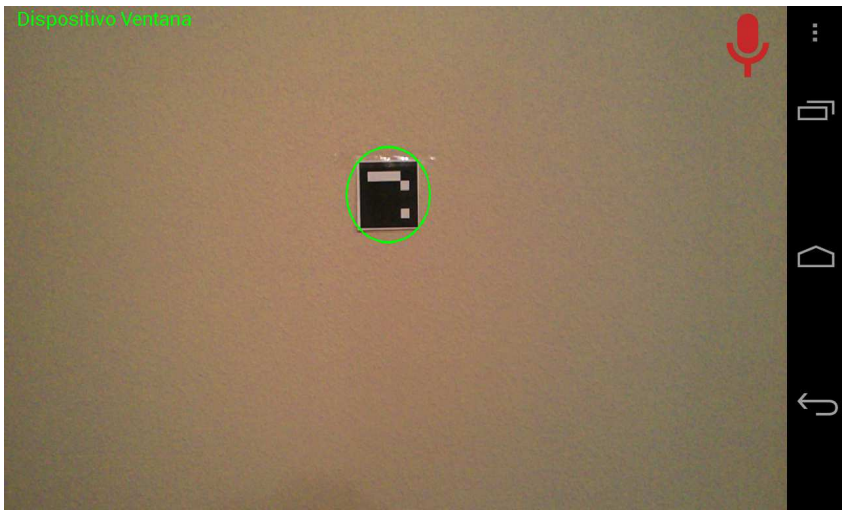


**Figura 12-11: Reconocimiento de voz**

12. Al pulsar el menú de opciones se muestra los distintos tipos de interfaz a utilizar.

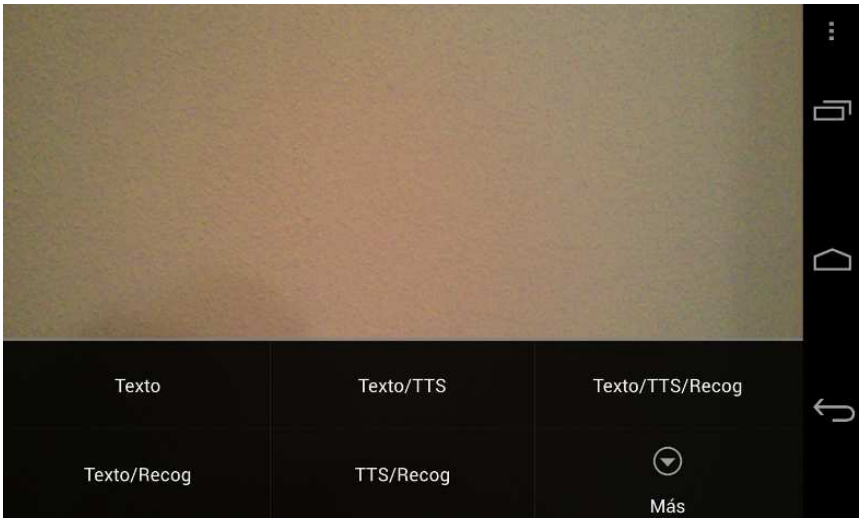


**Figura 12-12: Menú de opciones tipo de interfaz**

13. Resto de opciones del menú, que se muestrán al pulsar la opción Más (ver imagen anterior).


**Figura 12-13: Resto de opciones**

14. La Opción Tiempo Marcador establece el tiempo necesario para la cuenta atrás y poder seleccionar un marcador. Entre 1 y 5 segundos.


**Figura 12-14: Opción para establecer el tiempo del marcador**

15. Opción para configurar los datos del servidor de domótica, la dirección IP y el puerto.


**Figura 12-15: Configuración del servidor de domótica**

# 13. ANEXO C: GUÍA DE DESPLIEGUE

## 13.1 DESCARGA DESDE SVN

A continuación se muestran las instrucciones necesarias para poder desplegar completamente el sistema desde el código fuente a través del repositorio de SVN creado para el proyecto.

1. Abrir el entorno de desarrollo Eclipse.

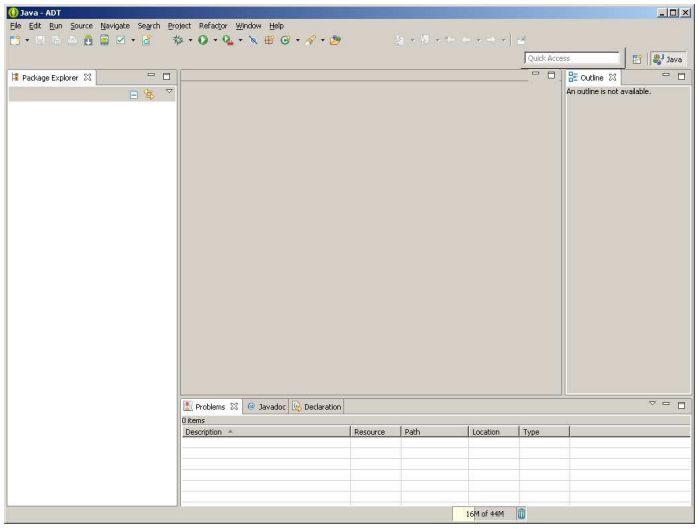


**Figura 13-1: IDE Eclipse**

2. Botón derecho sobre el área de proyectos: *Import*

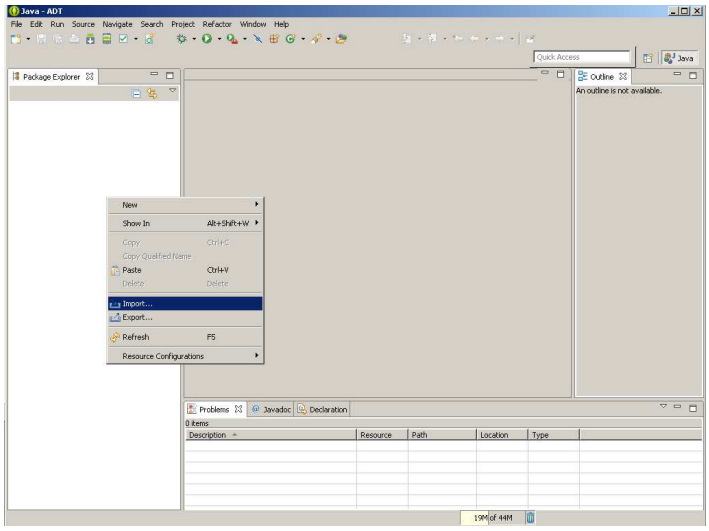


**Figura 13-2: Importar proyecto**

3. Proyecto desde el SVN (Eclipse Subversive)

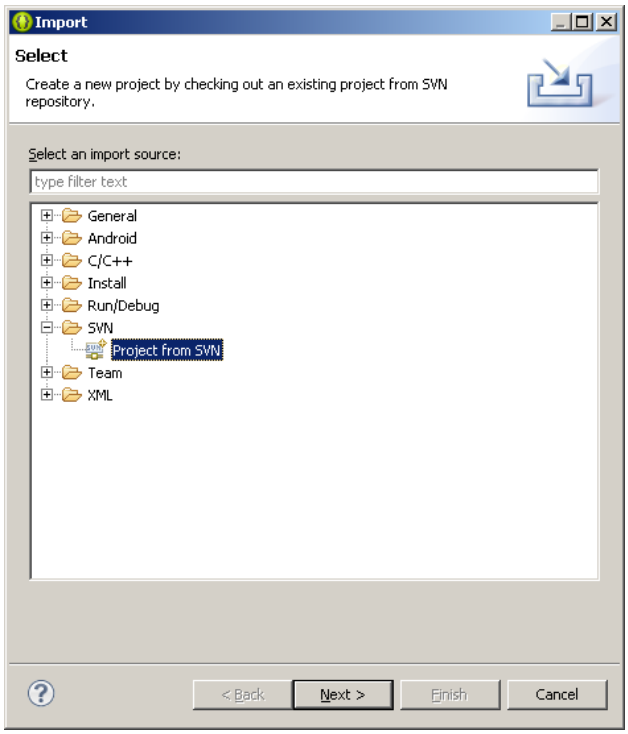


**Figura 13-3: Importar proyecto desde SVN**

4. URL hacia el *trunk* del proyecto en el repositorio:

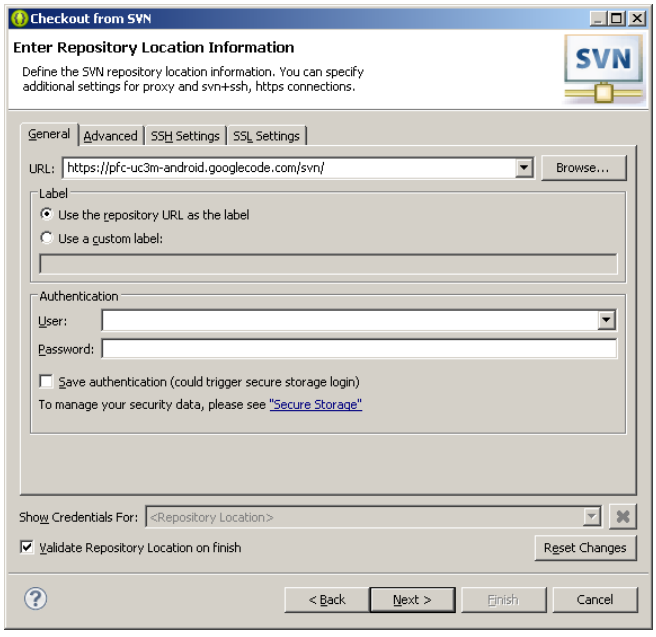


**Figura 13-4: URL hacia el repositorio**

5. Head Revision:

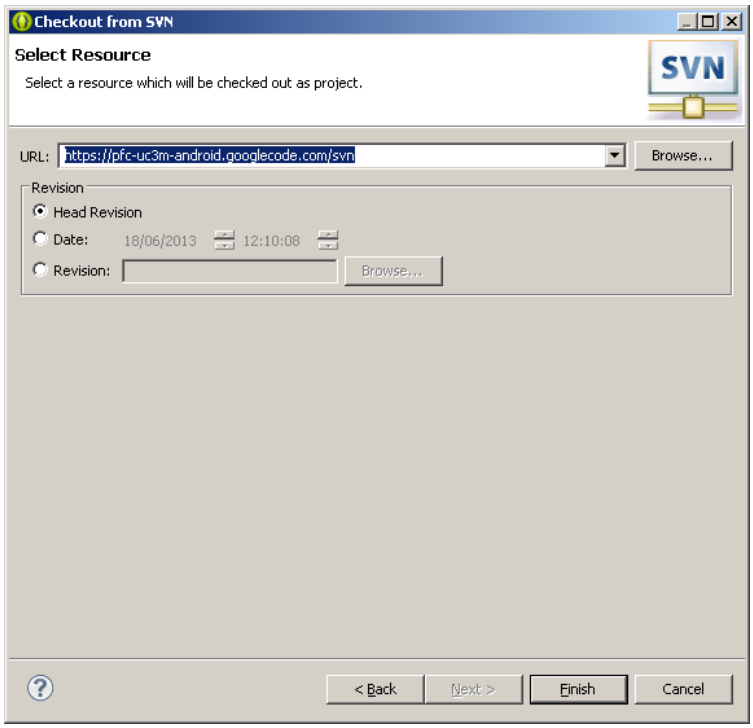


**Figura 13-5: Revision head**

6. Seleccionar la opción “Find projects…”:

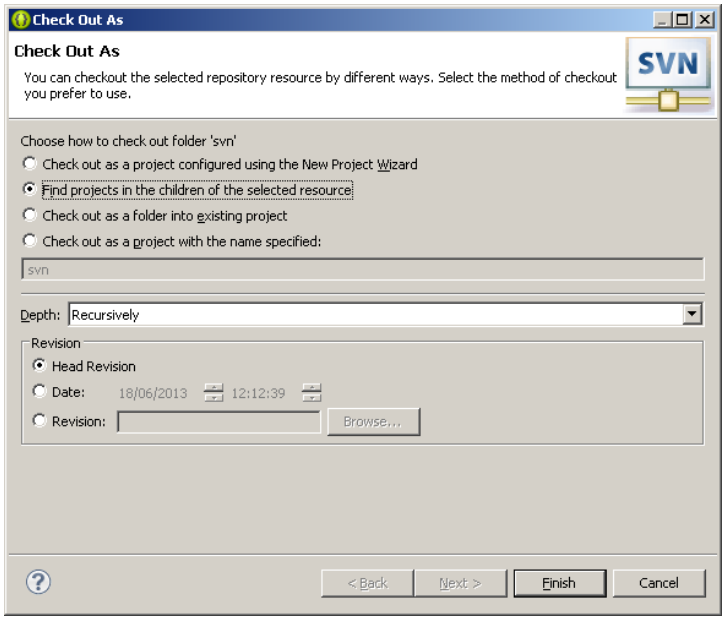


**Figura 13-6: Opción de Checkout**

7. Diálogo de progreso:

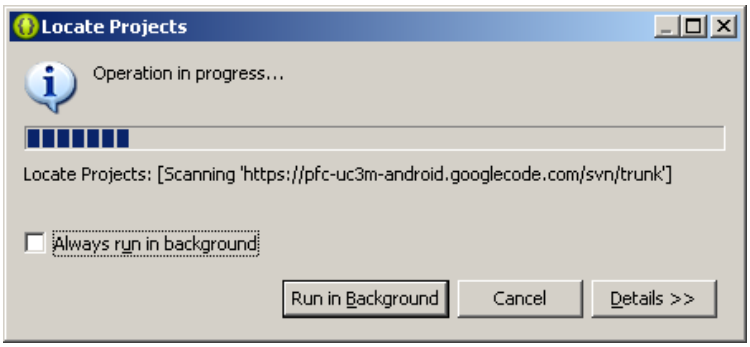


**Figura 13-7: Progreso búsqueda proyectos**

8. Seleccionar todos los proyectos:

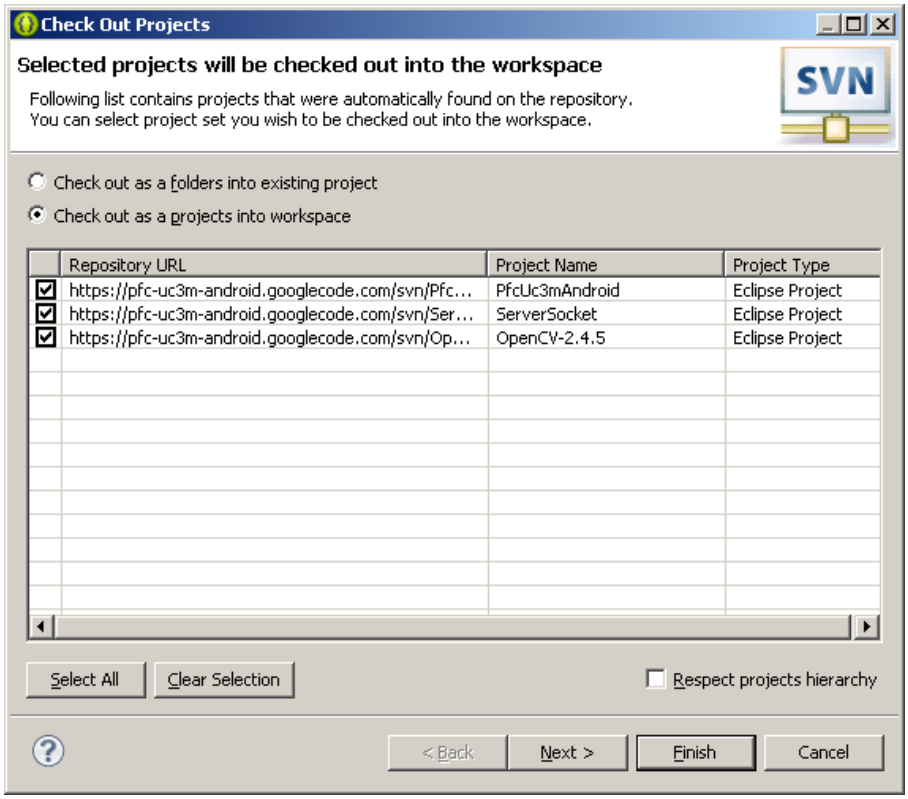


**Figura 13-8: Proyectos en el repositorio**

9. Cuadro de diálogo con la operación de *check out*:

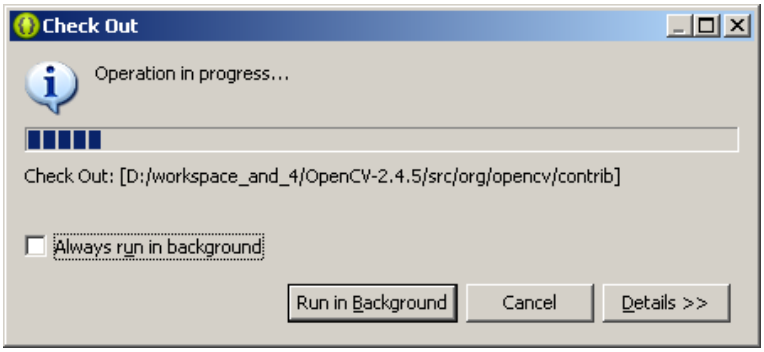


**Figura 13-9: Progreso de descarga**

10. Proyectos descargados en la vista de proyectos de Eclipse:

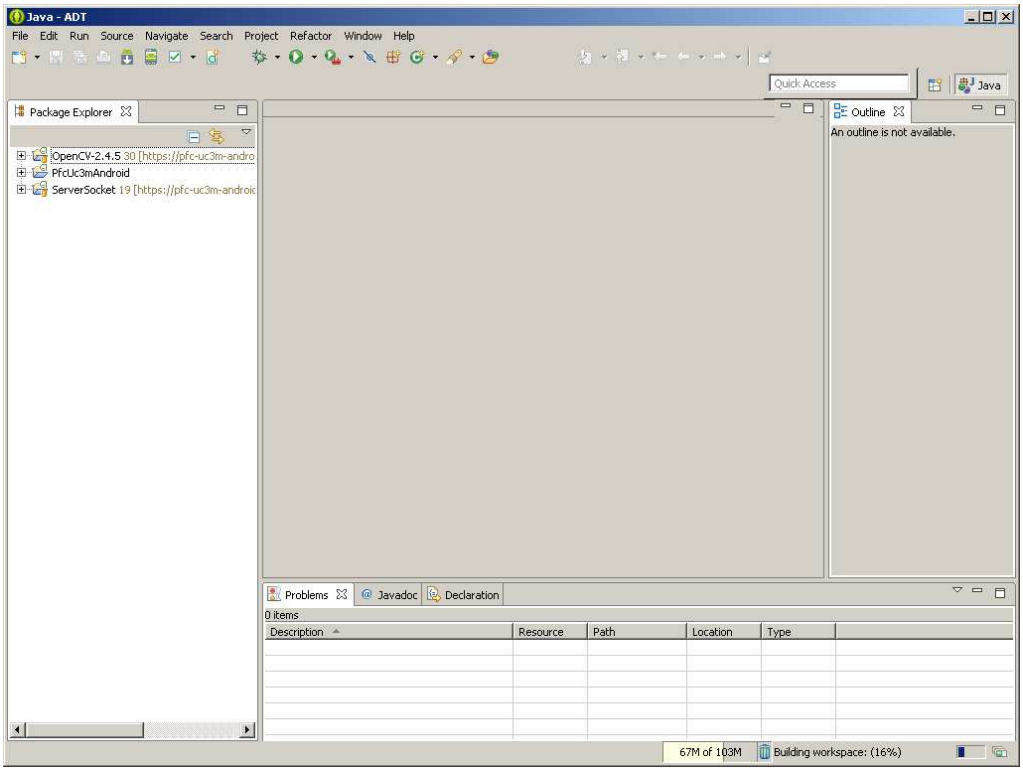


**Figura 13-10: Proyectos en el workspace**

11. Consola con el deployment del apk:

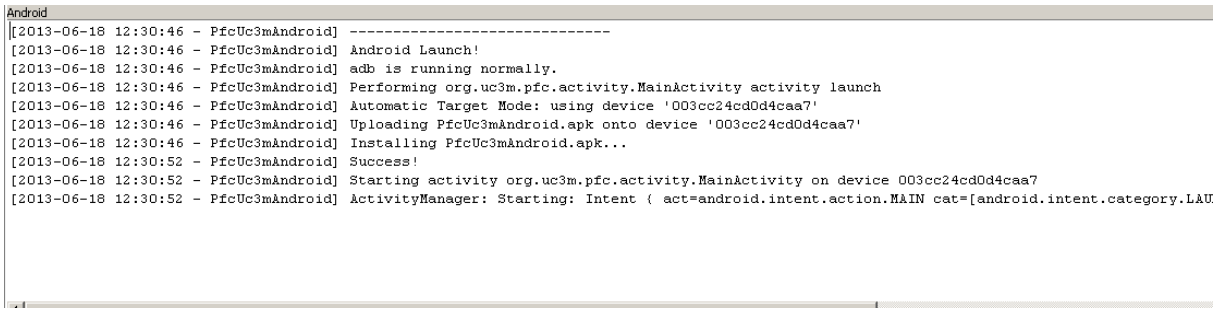


**Figura 13-11: Consola con el deployment del apk**

12. Logcat con el deployment del apk:

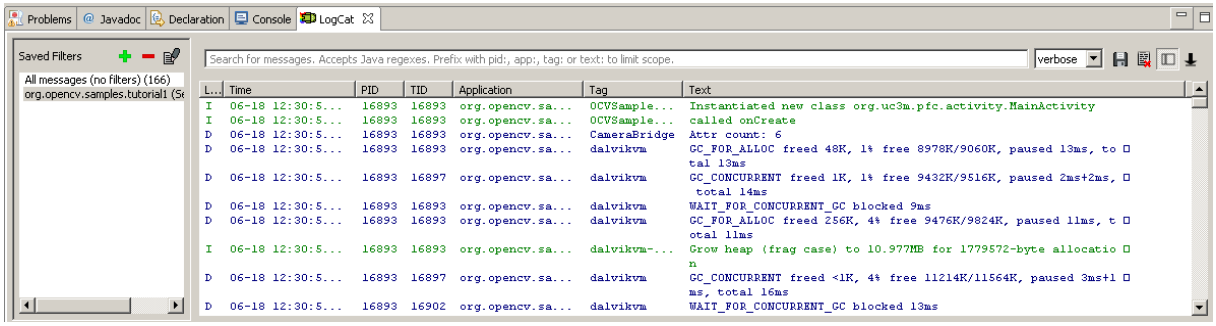


**Figura 13-12: Logcat con el deployment del apk 1**

13. Logcat con el deployment del apk:

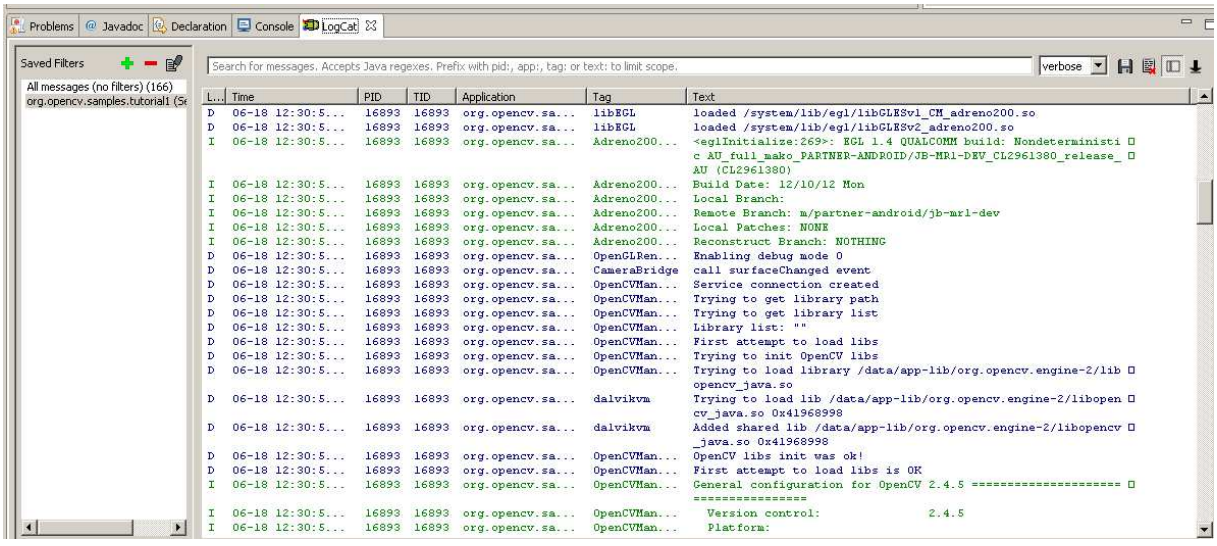


**Figura 13-13: Logcat con el deployment del apk 2**

## 13.2IMPORTACIÓN DESDE UN FICHERO ZIP

A continuación se muestra el proceso de importar el código fuente del proyecto desde el fichero comprimido con las fuentes:

1. Extraer el contenido del fichero zip al directorio del *workspace*.

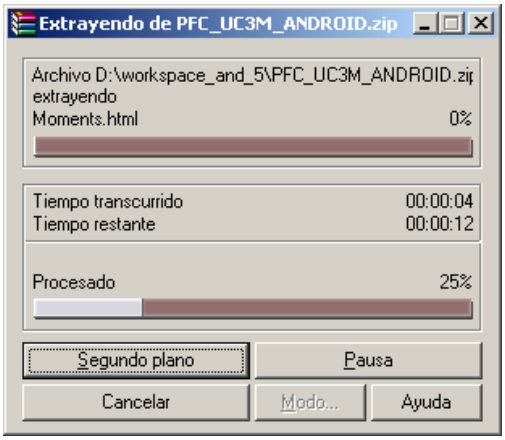


**Figura 13-14: Diálogo de extracción**

2. Listado de proyectos en el workspace:

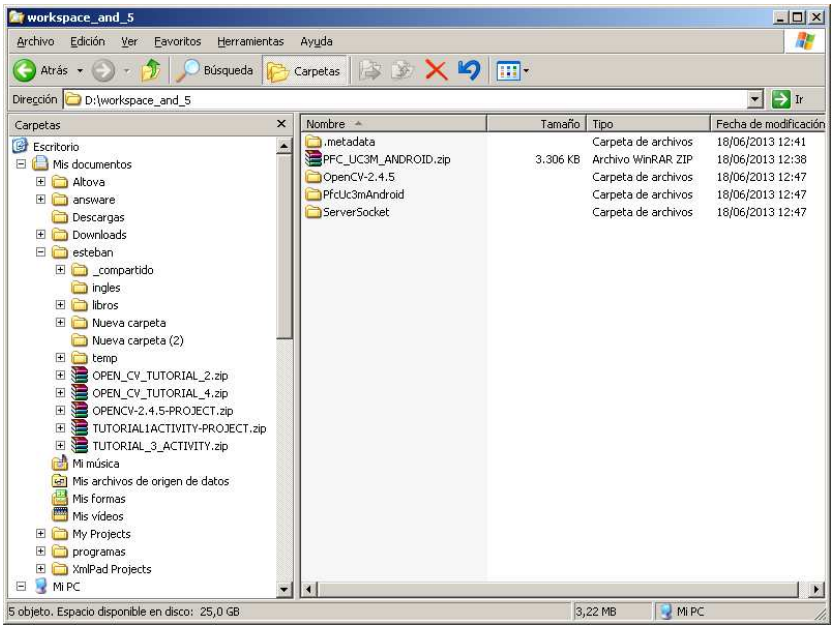


**Figura 13-15: Listado de proyectos workspace**

3. Importar proyectos existentes en el workspace:

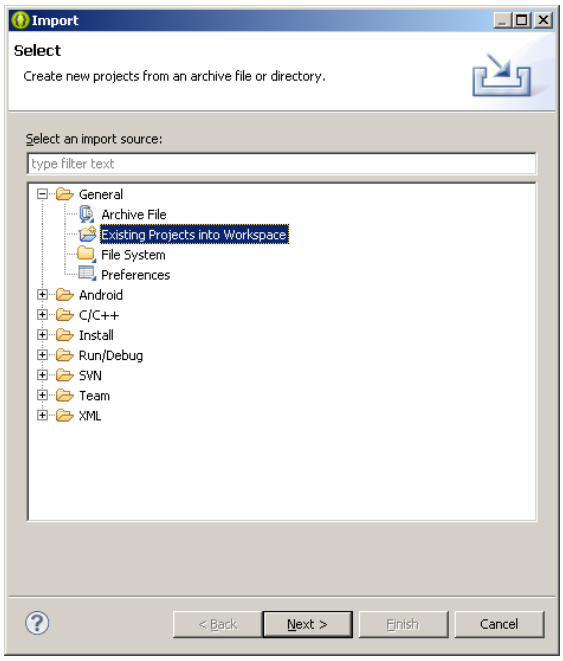


**Figura 13-16: Importar proyectos workspace**

4. Diálogo de selección con la vista de proyectos:

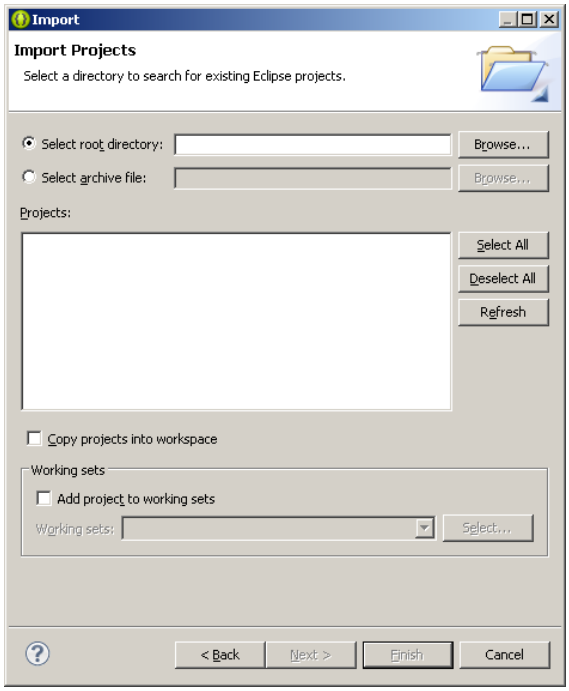


**Figura 13-17: Seleccionar de proyectos**

5. Selección del directorio:

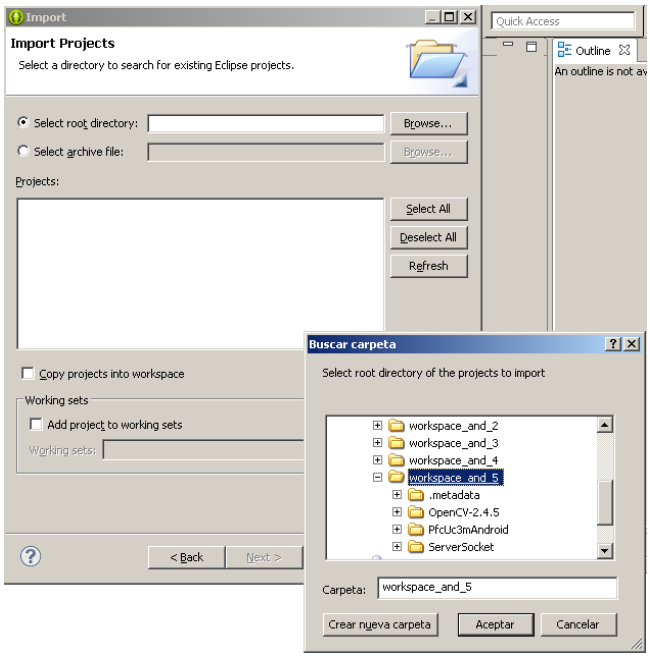


**Figura 13-18: Selección del directorio raiz**

6. Listado de proyectos seleccionados:

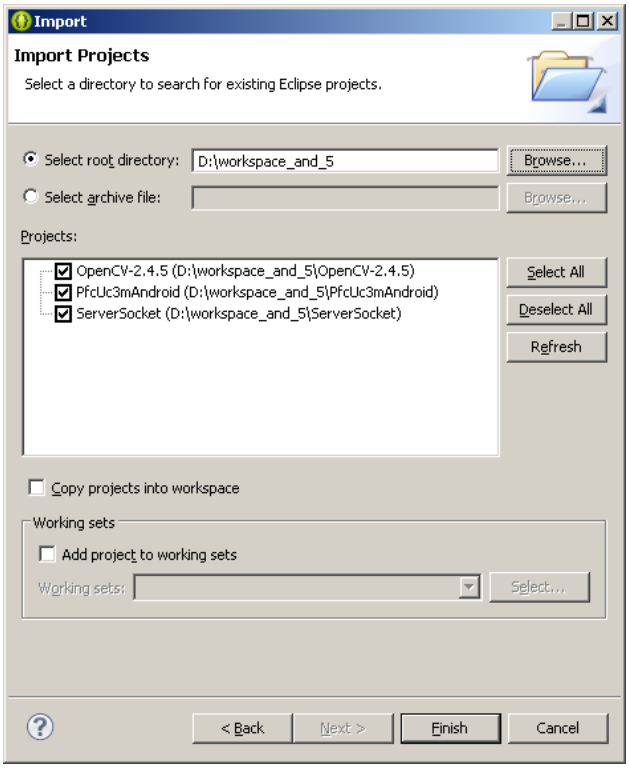


**Figura 13-19: Lista de proyectos seleccionados**

7. Lista de proyectos en el Eclipse:

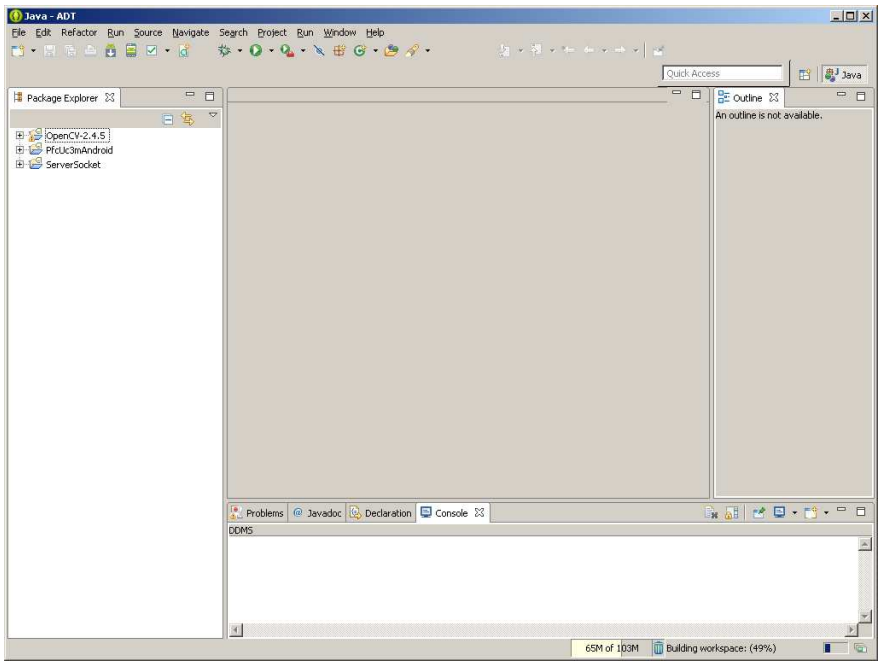


**Figura 13-20: Proyectos en Eclipse**

## 13.3EXPORTAR A UN .APK

A continuación se muestra el proceso de exportar el código fuente del proyecto a un fichero binario de android .apk

1. Sobre el proyecto *PfcUc3mAndroid*, botón derecho y elegir la opción *Export*.

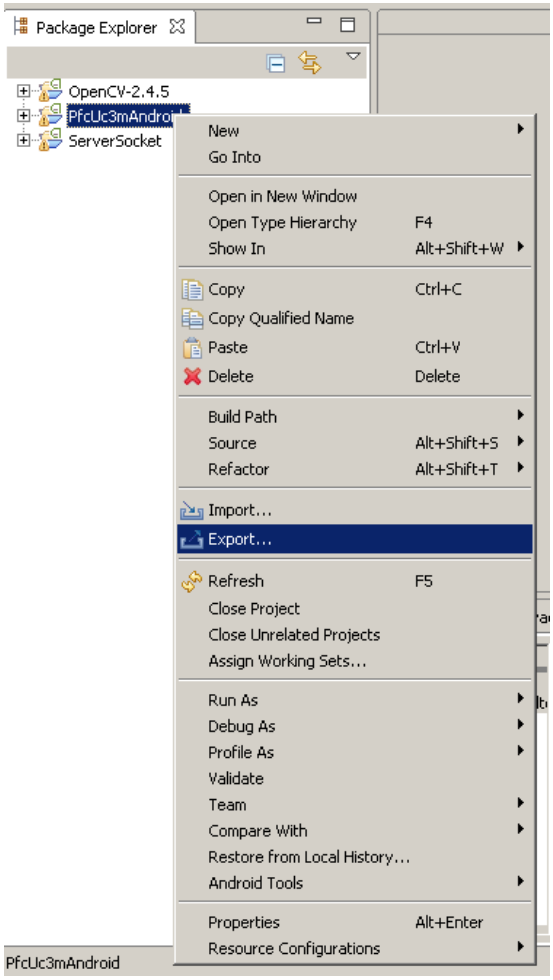


**Figura 13-21: Exportar proyecto**

2. Elegir la opción Android → Exportar Aplicación de Android:

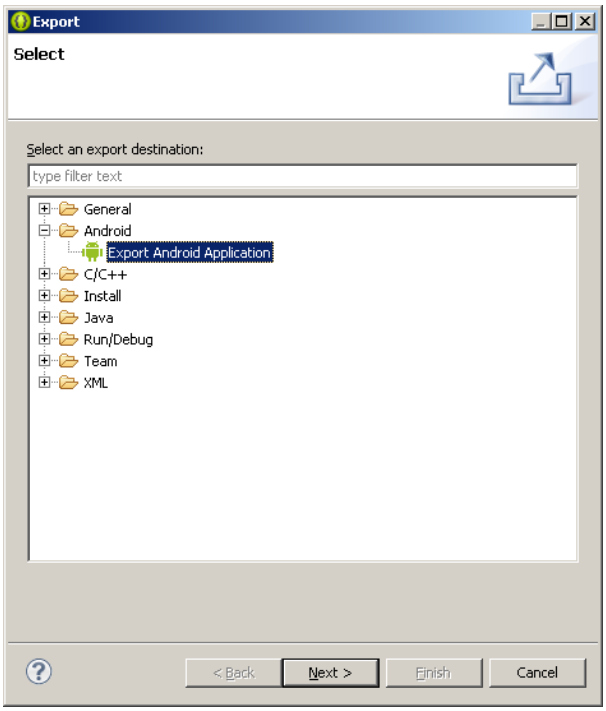


**Figura 13-22: Exportar aplicación android**

3. Seleccionar proyecto por defecto a exportar PfcUc3mAndroid.

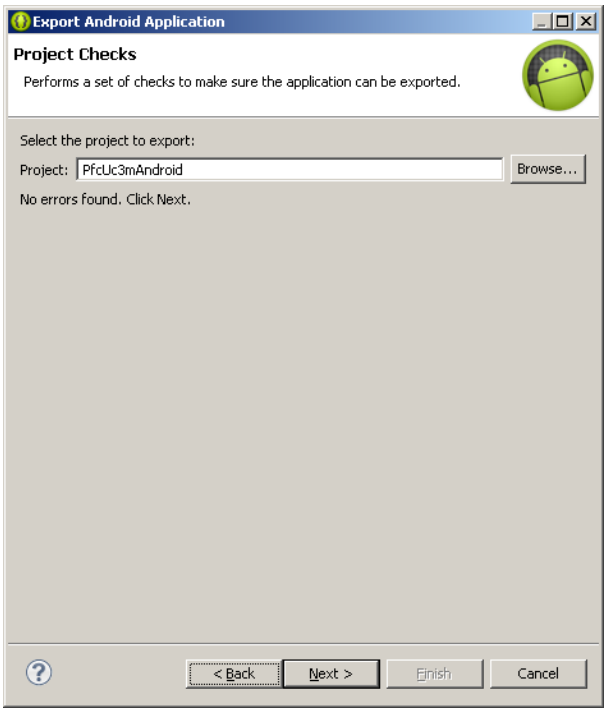


**Figura 13-23: Proyecto a exportar**

4. Generar un nueva clave o seleccionar una ya existente para el apk.

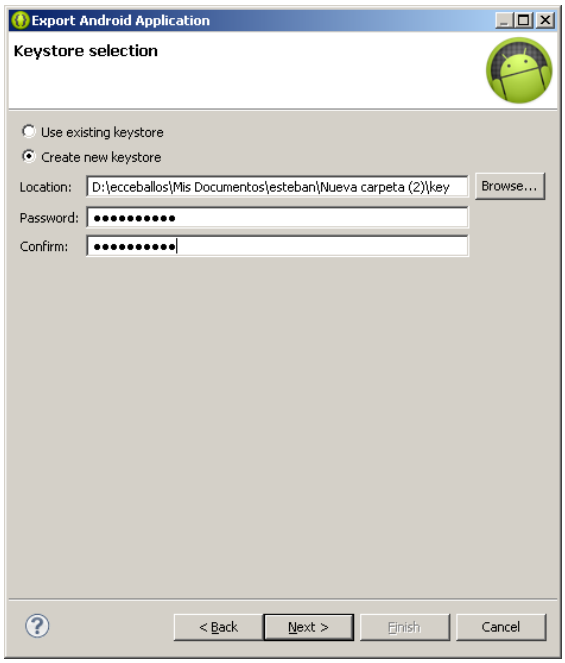


**Figura 13-24: Clave de seguridad para el apk**

5. Creación de una nueva clave para el apk.

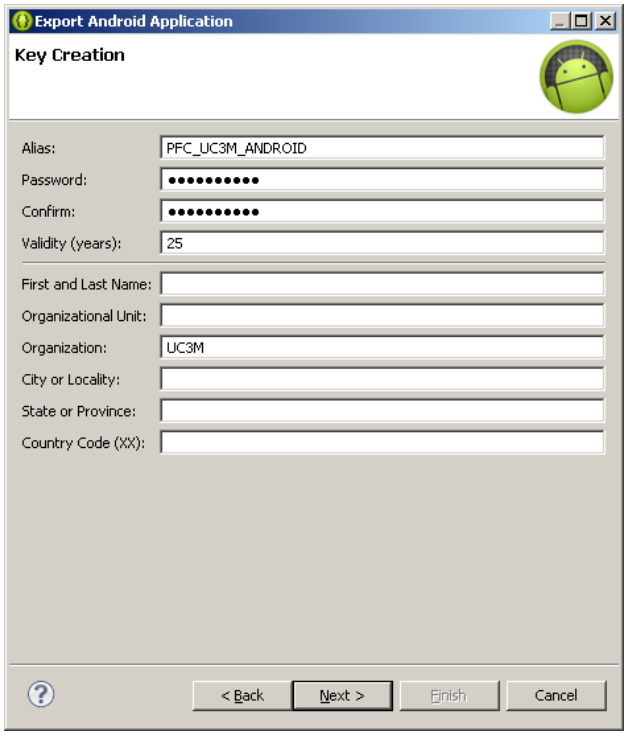


**Figura 13-25: Creación de una clave para el ap**

6. Selección de la carpeta para guardar el apk.

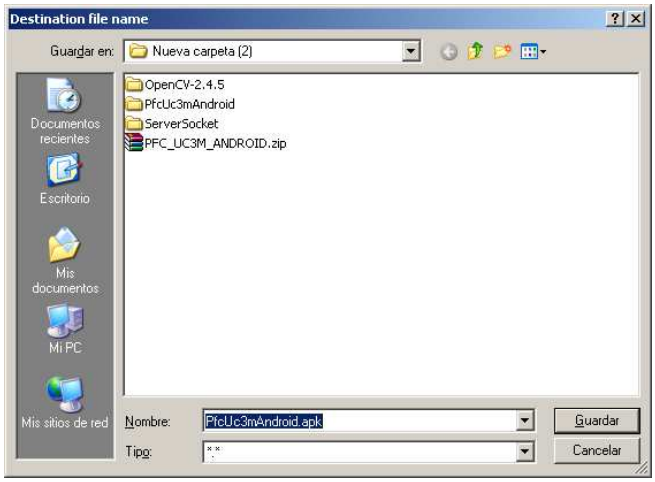


**Figura 13-26: Diálogo para guardar el apk**

7. Directorio destino del apk.

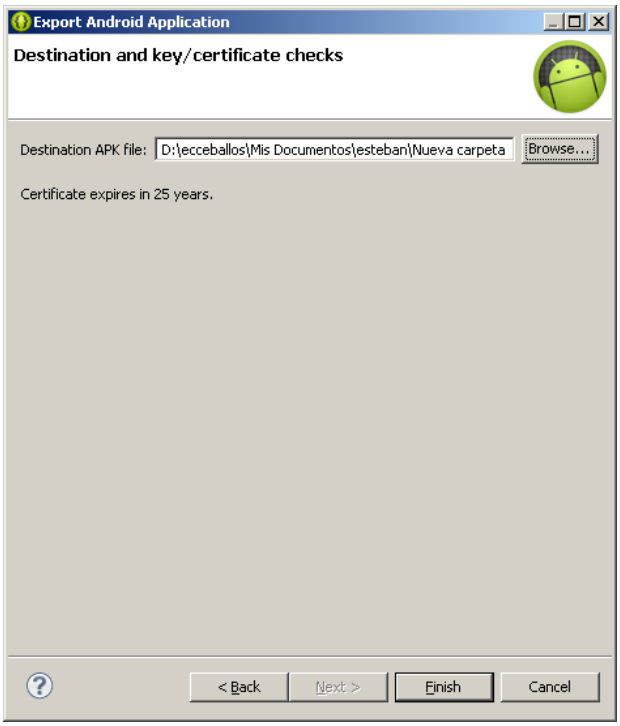


**Figura 13-27: Directorio destino del apk**

8. Carpeta con el apk final generado.

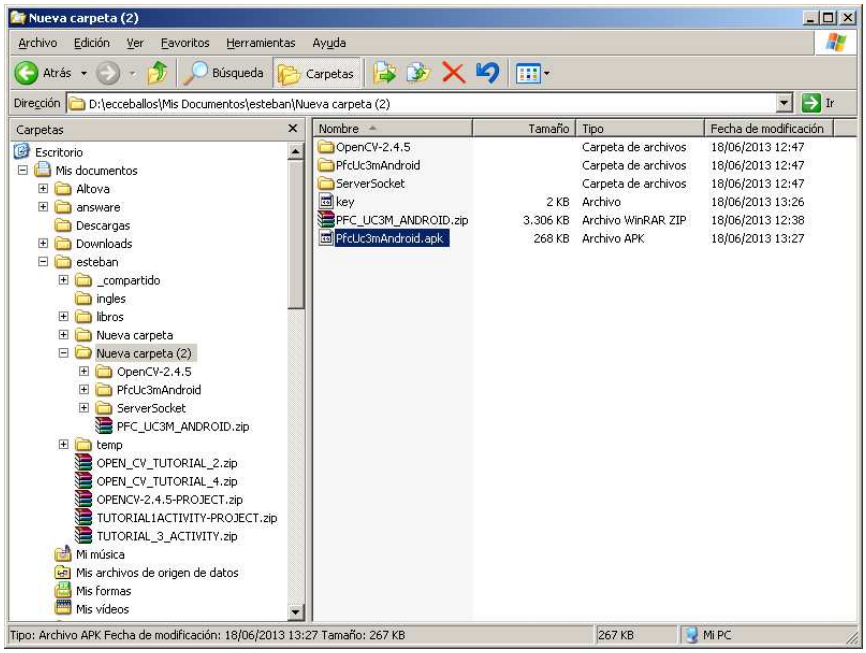


**Figura 13-28: Carpeta con el apk final generado**